The Name-Passing Calculus

Yuxi Fu Han Zhu
BASICS, Shanghai Jiao Tong University

Abstract

Name-passing calculi are foundational models for mobile computing. Research into these models has produced a wealth of results ranging from relative expressiveness to programming pragmatics. The diversity of these results call for clarification and reorganization. This paper applies a model independent approach to the study of the name-passing calculi, leading to a uniform treatment and simplification. The technical tools and the results presented in the paper form the foundation for a theory of name-passing calculus.

1 Mobility in Practice and in Theory

Mobile calculi feature the ability to pass around objects that contain channel names. Higher order CCS [91, 92, 93, 94] for instance, is a calculus with a certain degree of mobility. In a mobile calculus, a process that receives an object may well make use of the names which appear in the object to engage in further interactions. It is in this sense that the communication topology is dynamic. It was soon realized that the communication mechanism that restricts the contents of communications to the channel names gives rise to a simple yet versatile model that is more powerful than the process-passing calculi [79, 80, 82, 81]. This is the π-calculus of Milner, Parrow and Walker [60]. See [71] for a gentle introduction to the model and the history of the name-passing calculus and [87] for a broader coverage. A seemingly innocent design decision of the π-calculus is to admit a uniform treatment of the names. This decision is however not supported by the semantics of the mobile calculi. From a process term T one could construct the input prefix term

$$a(x).T \tag{1}$$

and the localization term

$$(x)T. \tag{2}$$

According to the definition of the π-calculus, the semantics of x which appears in (1) is far different from that of x in (2). In the former x is a name variable, or a dummy name, that can be instantiated by an arbitrary name when the prefix engages in an interaction. In the latter x is a local name that can never be confused with another name. The input prefix forces the unbound name x in T to be a name variable, whereas the localization operator forces the unbound name x in T to be a constant name. This apparent contradiction is behind all the semantic complications of the π-calculus. And nothing has been gained by these complications. In what follows we take a look at some of the issues caused by the confusion.

To begin with, the standard operational semantics of the π-calculus has not been very smooth. An extremely useful command in both practice and theory is the two leg if-statement *if* φ *then* S *else* T. In mobile calculi this can be defined by introducing the conditional terms $[x{=}y]T$ and $[x{\neq}y]T$. The semantics of these terms have been defined respectively by the match rule

$$\frac{T \xrightarrow{\lambda} T'}{[x{=}x]T \xrightarrow{\lambda} T'} \tag{3}$$

and the mismatch rule

$$\frac{T \xrightarrow{\lambda} T'}{[x{\neq}y]T \xrightarrow{\lambda} T'}\ x \neq y. \tag{4}$$

Rule (4) is unusual since it has an unusual side condition. How should we understand the side condition $x \neq y$? If x, y were constant names, the side condition of (4) would be pointless because $x \neq y$ would be evaluated to logical truth. The reason that (4) is necessary is precisely because x, y cannot be understood as constant names in the uniform treatment of the names. The correct reading of (4) is that "$[x{\neq}y]T \xrightarrow{\lambda} T'$ is admissible under the logical assumption $x \neq y$". It should now be clear that the popular semantics fails to support the following equivalence

$$T = [x{=}y]T + [x{\neq}y]T. \tag{5}$$

To see this, let $=$ be $\sim$, the strong early bisimilarity of Milner, Parrow and Walker [60]. According to the definition, $T \sim [x{=}y]T + [x{\neq}y]T$ if and only if $T\sigma \dot{\sim} ([x{=}y]T + [x{\neq}y]T)\sigma$ for every substitution σ, where $\dot{\sim}$ is the strong ground bisimulation equivalence. If σ is the identity substitution, it boils down to establishing the following equivalence

$$T \dot{\sim} [x{=}y]T + [x{\neq}y]T. \tag{6}$$

We may prove (6) under the assumption $x \neq y$. But we cannot prove (6) under the assumption $x = y$ since rule (3) does not allow us to do that. A related mistake is to introduce a boolean evaluation function $beval(\_)$ whose inductive definition includes the following clauses:

$$\begin{aligned} beval(x{=}y) &\stackrel{\text{def}}{=} \bot, \\ beval(x{\neq}y) &\stackrel{\text{def}}{=} \top. \end{aligned}$$

This would lead to the axioms

$$\begin{aligned} [x{=}y]T &= \mathbf{0}, \\ [x{\neq}y]T &= T, \end{aligned}$$

which are wrong if observational equivalences are closed under prefix operations. One way to rectify (3) is to introduce the following

$$\frac{T \xrightarrow{\lambda} T'}{[x{=}y]T \xrightarrow{\lambda} T'}\ x = y. \tag{7}$$

The rule (7) does not completely eradicate the problem. Take for example the following instance of the expansion law

$$\overline{x}x \,|\, y(v) = \overline{x}x.y(v) + y(v).\overline{x}x + [x{=}y]\tau. \tag{8}$$

The right hand side of (8) can do a τ-action under the assumption $x = y$. However the operational semantics of the π-calculus does not admit a τ-action of the left hand side of (8) even if the logical assumption $x = y$ is made. One solution to the above problem is to introduce the following rule

$$\frac{T\{y/x\} \xrightarrow{\lambda\{y/x\}} T'\{y/x\}}{T \xrightarrow{\lambda} T'}\ \text{if } x = y \text{ and } bn(\lambda) \cap \{x, y\} = \emptyset.$$

This rule is necessary because the terms $T\{y/x\}, T$ are distinct in the meta logic. Our scrutiny of the match/mismatch semantics takes us to the symbolic approach of Hennessy and Lin [36, 45, 46, 47, 48, 49]. In the symbolic semantics, one has that $[x{\neq}y]T \xrightarrow{x\neq y,\lambda} T'$, meaning that the action is admissible under the logical assumption $x \neq y$. Similarly one has $[x{=}y]T \xrightarrow{x=y,\lambda} T'$. Notice that this transition is very different from the transition $[x{=}x]T \xrightarrow{\top,\lambda} T'$. In the symbolic approach the action $T \xrightarrow{\top,\lambda} T'$ is simulated by the combined effect of $[x{\neq}y]T \xrightarrow{x\neq y,\lambda} T'$ and $[x{=}y]T \xrightarrow{x=y,\lambda} T'$, not by any single action sequence of $[x{=}y]T + [x{\neq}y]T$. If we think of it, the symbolic semantics not only provides the correct labeled transition semantics upon which we may study the observation theory of the π-calculus, but also makes clear the problem caused by the confusion of the name variables and the names.

Secondly the observational theory of mobile processes is made more complex than it is. One well-known phenomenon is that, unlike in CCS [56, 61], some standard definitions of process equivalence lead to different equality

relations. The standard definition of bisimulation gives rise to ground bisimilarity that is not closed under input prefix operation. The solution proposed in [60] is to take the substitution closure. The resulting relation is the early equivalence. If substitution closure is required in every bisimulation step, one obtains Sangiorgi's open bisimilarity [83]. The open bisimilarity is strictly finer than the early equivalence, which is in turn much finer than the ground bisimilarity. The open bisimilarity can be further improved to quasi open bisimilarity [86], which lies nontrivially between the open bisimilarity and the early equivalence. The barbed equivalence can be defined by placing substitution closure at the beginning of bisimulations, which gives rise to the equivalence studied by Milner and Sangiorgi [61]. It can also be defined by requiring that every bisimulation step should be closed under substitutions of names. It is shown by Sangiorgi and Walker [86] that the latter coincides with the quasi open bisimilarity. It is easy to see that the barbed equivalence is weaker than the early equivalence. It is not yet clear however if it is subsumed by the early equivalence. Putting aside the issue of which of these equivalences is more authoritative than the rest, we would like to point out that the substitution closure requirement is an algebraic requirement rather than an observational one. From the true spirit of the observation theory, an environment can never detect any difference between $a(x).\overline{b}c + \overline{b}c.a(x)$ and $a(x) \mid \overline{b}c$, since it can never force the distinct names a, b to be equal. This issue of reconciling the inconsistency between the observational view and the algebraic view must be addressed to achieve a better theory of the mobile processes.

The algebraic requirement also makes the testing theory of mobile processes hard to comprehend. In the testing theory developed by De Nicola and Hennessy [19], the behaviors of a process are judged by testers. Two processes are testing equivalent if no testing can detect any behavioral difference between them. Like the bisimulation approach, the testing approach fails to give rise to a reasonable equivalence on the mobile processes. In order to respect the name uniformity and obtain a useful equivalence at the same time, the algebraic condition must be imposed. See [9] for more on this issue. In some sense the substitution closure condition completely defeats the philosophy of the testing theory.

In retrospect, the confusion of the names and the name variables is not out of the desire to model mobility, since mobility can be achieved by using the name variables anyway. If channels have a physical existence, computations or interactions really should not manipulate channels. What they are supposed to do is to make use of the channels for the purpose of interaction. According to this interpretation, all channel names ought to be constant. To model mobility, the introduction of a dichotomy between the names and the name variables is not only an obvious choice, it is the only choice. The variables are there for mobility.

In theory of expressiveness, the name dichotomy provides a basis for comparing the relative expressive powers of calculi. The straightforward translation from CCS to the π-calculus for instance is fully abstract if in the π-calculus a line is drawn between the names and the name variables. The translation takes the equivalent CCS processes, say $a \mid \overline{b}$ and $a.\overline{b} + \overline{b}.a$, to the equivalent π-processes $a(x) \mid \overline{b}(y)$ and $a(x).\overline{b}(y) + \overline{b}(y).a(x)$. If the names are treated uniformly, the target model would have a much stronger process equality than the source model. In such a framework it is not even clear if a reasonably good fully abstract translation from CCS to the π-calculus exists. Other expressiveness results can also be best interpreted using the name dichotomy. Sangiorgi-Thomsen's encoding of the higher order CCS in the π-calculus is another example. The process variables of the higher order CCS are translated to the name variables of the π-calculus, while the names of the former *are* the names of the latter. This encoding is shown to be fully abstract by Sangiorgi [79, 80]. Again if the names of the π-calculus are treated uniformly, the encoding would not even be sound. We could give more examples to support the proposition that a dichotomic understanding should be preferred. But the point is already made. The names play a universal role in process theory. Without the assumption that all names are constant, expressiveness results about process calculi are bound to be chaotic [64].

When applying the mobile calculi to interpret programming phenomena, the name dichotomy has always been enforced. It is sufficient to give just one example. An early work was done by Walker [103, 104], who defined the operational semantics of an object oriented language in terms of the operational semantics of the π-calculus. The idea of the interpretation can be summarized as follows. An object is modeled by a prefix process of the form $obj n(x).O$, where $obj n$ is the name of the object. A method is interpreted as a replicated process of the form $!mthd(z).M$, where *mthd* is the method name. The method can be invoked by a process of the form $\overline{mthd}(v).P$ that supplies the value v to the method parameter. Without going into details, it is already obvious that for this interpretation to make sense, it is important to maintain a distinction between the names and the name variables. We could give many other applications of the mobile calculi. But it suffices to say that in all these applications, there is a clear cut distinction between the names and the name variables.

The above discussions lead to the conclusion that, for both theoretical and practical reasons, the π-calculus should be defined using the name dichotomy. The dichotomy has been introduced in literature using type systems. If one thinks of the type of a channel name as defining the interface property of the channel, then the type theoretical solution does not seem appropriate since the difference between a name and a name variable is not about interface property. It is our view that the issue should be treated at a more fundamental level.

This is both a survey paper and research paper. Since we adopt a new uniform and simplified presentation of the π-calculus, there are technical contributions throughout the paper. In view of the huge literature on the π-calculus [87], it is a daunting task to give an overall account of the various aspects of the model. Our strategy in this paper is to present the foundational core of the π-calculus, covering the observational theory, the algebraic theory and the relative expressiveness. The novelty of our treatment is that, by applying a model independent approach throughout the paper, a great deal of simplification and unification are achieved. Our effort can be summarized as follows:

- We show that a concise operational semantics of π is available.
- We demonstrate that the observational theory of π is far less diverse than it has been perceived.
- We point out that the algebraic theory of π is simpler than has been suggested.

The above claims are supported by the following technical contributions:

- A general model independent process equality, the absolute equality, is applied to the π-calculus. It is proved that the well known bisimulation equivalences of the π-calculus, mentioned in this introduction, either coincide with a weak version of the absolute equality or can be safely ignored.
- A model independent equivalence, the box equality, is defined and applied to the π-calculus. It is demonstrated that this new equivalence coincides with the well known rectification of the testing equivalence in the π-calculus.
- Two complete proof systems for the set of the finite π-processes are presented, one for the absolute equality, the other for the box equality.

The model independent theory of process calculi is systematically developed in [27]. In particular the absolute equality and the subbisimilarity used in this paper are taken from [27]. It should be pointed out however that the present paper has been made self-contained.

Most of the lemmas are stated without proof. A well-informed reader would have no problem in supplying the proof details.

The rest of the paper is organized into five sections. Section 2 defines our version of the π-calculus. Section 3 studies the model independent observation theory of the π-calculus. Section 4 discusses the relative expressiveness of some well known variants of the π-calculus. Section 5 presents a uniform account of the proof systems for the finite π-processes. Section 6 points out how a theory of the π-calculus can be developed using the framework set up in this paper.

2 Pi Calculus

We assume that there is an infinite countable set $\mathcal{N}$ of *names*, an infinite countable set $\mathcal{N}_v$ of *name variables*. These sets will be ranged over by different lower case letters. Throughout the paper the following conventions will be enforced:

- The set $\mathcal{N}$ is ranged over by a, b, c, d, e, f, g, h.
- The set $\mathcal{N}_v$ is ranged over by u, v, w, x, y, z.
- The set $\mathcal{N} \cup \mathcal{N}_v$ is ranged over by l, m, n, o, p, q.

A name variable acts as a place holder that need be substantiated by a name. By its very nature, a name variable cannot be used as a channel for interaction. Similarly it cannot be used as a message passed around in a communication.

2.1 Process

To give a structural definition of processes, we need to introduce terms. The set $\mathcal{T}$ of *π-terms* is inductively generated by the following BNF:

$$S, T \quad := \quad \mathbf{0} \mid \sum_{i\in I} n(x).T_i \mid \sum_{i\in I} \overline{n}m_i.T_i \mid S \mid T \mid (c)T \mid [p{=}q]T \mid [p{\neq}q]T \mid !\pi.T,$$

where I is a finite nonempty indexing set and

$$\pi \quad := \quad n(x) \mid \overline{n}m.$$

Here $n(x)$ is an input prefix and $\overline{n}m$ an output prefix. The *nil* process $\mathbf{0}$ cannot do anything in any environment. For each $i \leq n$, the component $n(x).T_i$ is a *summand* of the *input choice* term $\sum_{i\in I} n(x).T_i$, where the name variable x is *bound*. A name variable is *free* if it is not bound. Similarly the component $\overline{n}m_i.T_i$ is a summand of the *output choice* term $\sum_{i\in I} \overline{n}m_i.T_i$. Notice that input and output choices are syntactically simpler than the separated choices [67]. The term $T \mid T'$ is a concurrent *composition*. The restriction $(c)T$ is in *localization* form, where the name c is *local*. A name is *global* if it is not local. The following functions will be used.

- $gn(\_)$ returns the set of the global names.
- $ln(\_)$ returns the set of the local names.
- $n(\_)$ returns the set of the names.
- $fv(\_)$ returns the set of the free name variables.
- $bv(\_)$ returns the set of the bound name variables.
- $v(\_)$ returns the set of the name variables.

The guard $[p{=}q]$ is a *match* and $[p{\neq}q]$ a *mismatch*. The term $!\pi.T$ is a *guarded replication* and '!' a replication operator. The guarded replication is equivalent to the general replication of the form $!T$. The transformation from the general replication to the guarded replication makes use of an auxiliary function $(\_)^c$ defined on the replication free terms. The structural definition is as follows.

$$\begin{aligned}
(\mathbf{0})^c &\stackrel{\text{def}}{=} \mathbf{0},\\
(\pi.T)^c &\stackrel{\text{def}}{=} \pi.(\overline{c}c \mid T^c),\\
(T_1 \mid T_2)^c &\stackrel{\text{def}}{=} T_1^c \mid T_2^c,\\
((a)T)^c &\stackrel{\text{def}}{=} (a)T^c,\\
([p{=}q]T)^c &\stackrel{\text{def}}{=} [p{=}q]T^c,\\
([p{\neq}q]T)^c &\stackrel{\text{def}}{=} [p{\neq}q]T^c.
\end{aligned}$$

If neither c nor z is in T, we may define $(!T)^c$ by the process $(c)(\overline{c}c \mid \overline{c}c \mid !c(z).T^c)$. It is clear that there would be no loss of expressive power if guarded replication is further restrained to the form $!p(x).T$ or $!\overline{p}q.T$.

A *finite* π-term is one that does not contain any replication operator. A π-term is *open* if it contains free name variables; it is *closed* otherwise. A closed π-term is also called a *π-process*. We write $\mathcal{P}$ for the set of the π-processes, ranged over by L, M, N, O, P, Q. Some derived prefix operators are defined as follows.

$$\begin{aligned}
\overline{n}(c).T &\stackrel{\text{def}}{=} (c)\overline{n}c.T,\\
n.T &\stackrel{\text{def}}{=} n(z).T \ \text{ for some } z \notin fv(T),\\
\overline{n}.T &\stackrel{\text{def}}{=} \overline{n}(c).T \ \text{ for some } c \notin gn(T),\\
\tau.T &\stackrel{\text{def}}{=} (c)(c.T \mid \overline{c}) \ \text{ for some } c \notin gn(T).
\end{aligned}$$

Furthermore we introduce the following polyadic prefixes:

$$n(x_1,\ldots,x_n).T \stackrel{\text{def}}{=} n(z).z(x_1).\cdots.z(x_n).T \text{ for some } z \notin fv(T),$$
$$\overline{n}\langle p_1,\ldots,p_n\rangle.T \stackrel{\text{def}}{=} \overline{n}(c).\overline{c}p_1.\cdots.\overline{c}p_n.T \text{ for some } c \notin gn(T),$$

where $n > 1$. These two derived operators make it clear how to simulate the polyadic π-calculus [59] in the (monadic) π-calculus.

Both bound name variables and local names are subject to α-conversion. Throughout the paper, it is assumed that α-conversion is applied whenever it is necessary to avoid confusion. This will be called the *α-convention*. In for example the structural composition rule to be defined later, the side conditions are redundant in the presence of the α-convention.

A *condition*, denoted by ϕ, φ, ψ, is a finite concatenation of matches and/or mismatches. The concatenation of zero match/mismatch is denoted by $\top$, and its negation is denoted by $\bot$. We identify syntactically $(\top)T$ with T and $(\bot)T$ with $\mathbf{0}$. If $\mathcal{F}$ is the finite name set $\{n_1,\ldots,n_i\}$, the notation $[p{\notin}\mathcal{F}]T$ stands for $[p{\neq}n_1]\ldots[p{\neq}n_i]T$.

A *renaming* is a partial injective map $\alpha : \mathcal{N} \rightharpoonup \mathcal{N}$ whose domain of definition $dom(\alpha)$ is finite. A *substitution* is a partial map $\sigma : \mathcal{N}_v \rightharpoonup \mathcal{N} \cup \mathcal{N}_v$ whose domain of definition $dom(\sigma)$ is finite. An *assignment* is a partial map $\rho : \mathcal{N}_v \rightharpoonup \mathcal{N}$ that associates a name to a name variable in the domain of ρ. It is convenient to extend the definition of an assignment ρ by declaring $\rho(a) = a$ for all $a \in \mathcal{N}$. Renaming, substitution and assignment are used in postfix. We often write $\{n_1/x_1,\ldots,n_i/x_i\}$ for a substitution, and similar notation is used for renaming. The notation $\rho[x \mapsto a]$ stands for the assignment that differs from ρ only in that $\rho[x \mapsto a]$ always maps x onto a whereas $\rho(x)$ might be different from a or undefined. Whenever we write $\rho(x)$ we always assume that ρ is defined on x.

An assignment ρ satisfies a condition ψ, denoted by $\rho \models \psi$, if $\rho(m) = \rho(n)$ for every $[m{=}n]$ in ψ, and $\rho(m){\neq}\rho(n)$ for every $[m{\neq}n]$ in ψ. We write $\rho \models \psi \Rightarrow \phi$ to mean that $\rho \models \phi$ whenever $\rho \models \psi$, and $\rho \models \psi \Leftrightarrow \phi$ if both $\rho \models \psi \Rightarrow \phi$ and $\rho \models \phi \Rightarrow \psi$. We say that ψ is *valid*, notation $\models \psi$ (or simply ψ), if $\rho \models \psi$ for every assignment ρ. A useful valid condition is the following.

$$(x{\notin}\mathcal{F}) \vee \bigvee_{n\in\mathcal{F}} (x{=}n), \tag{9}$$

where $\mathcal{F}$ is a finite subset of $\mathcal{N} \cup \mathcal{N}_v$. Given a condition φ, we write $\varphi^{=}$ and $\varphi^{\neq}$ respectively for the condition $\bigwedge\{m{=}n \mid m,n \in n(\varphi) \cup v(\varphi) \text{ and } \models \varphi \Rightarrow m{=}n\}$ and the condition $\bigwedge\{m{\neq}n \mid m,n \in n(\varphi) \cup v(\varphi) \text{ and } \models \varphi \Rightarrow m{\neq}n\}$.

2.2 Semantics

The semantics is defined by a labeled transition system structurally generated by a set of rules. The set $\mathcal{L}$ of *labels* for π-terms, ranged over by ℓ, is

$$\{ab, \overline{a}b, \overline{a}(c) \mid a,b,c \in \mathcal{N}\}$$

where $ab, \overline{a}b, \overline{a}(c)$ denote respectively an input action, an output action and a bound output action. The set $\mathcal{L}^*$ of the finite strings of $\mathcal{L}$ is ranged over by ℓ^*. The empty string is denoted by ϵ. The set $\mathcal{A} = \mathcal{L} \cup \{\tau\}$ of actions is ranged over by λ and its decorated versions. The set $\mathcal{A}^*$ of the finite strings of $\mathcal{A}$ is ranged over by λ^*. With the help of the action set, we can define the operational semantics of the π-calculus by the following labeled transition system.

Action

$$\frac{}{\sum_{i\in I} a(x).T_i \xrightarrow{ac} T_i\{c/x\}}\ i \in I \qquad \frac{}{\sum_{i\in I} \overline{a}c_i.T_i \xrightarrow{\overline{a}c_i} T_i}\ i \in I$$

Composition

$$\frac{T \xrightarrow{\lambda} T'}{S \mid T \xrightarrow{\lambda} S \mid T'} \qquad \frac{S \xrightarrow{ab} S' \quad T \xrightarrow{\overline{a}b} T'}{S \mid T \xrightarrow{\tau} S' \mid T'} \qquad \frac{S \xrightarrow{ac} S' \quad T \xrightarrow{\overline{a}(c)} T'}{S \mid T \xrightarrow{\tau} (c)(S' \mid T')}$$

Localization

$$\frac{T \xrightarrow{\overline{a}c} T'}{(c)T \xrightarrow{\overline{a}(c)} T'} \qquad \frac{T \xrightarrow{\lambda} T'}{(c)T \xrightarrow{\lambda} (c)T'}\ c \notin n(\lambda)$$

Condition

$$\frac{T \xrightarrow{\lambda} T'}{[a{=}a]T \xrightarrow{\lambda} T'} \qquad \frac{T \xrightarrow{\lambda} T'}{[a{\neq}b]T \xrightarrow{\lambda} T'}$$

Replication

$$\frac{}{!\overline{a}b.T \xrightarrow{\overline{a}b} T \mid !\overline{a}b.T} \qquad \frac{}{!a(x).T \xrightarrow{ab} T\{b/x\} \mid !a(x).T}$$

The first composition rule takes care of the structural property. The second and the third define interactions. Their symmetric versions have been omitted. Particular attention should be paid to the action rules. An input action may receive a name from another term. It is not supposed to accept a name variable. This is because an output action is allowed to release a name, not a name variable. To go along with this semantics of interaction, the rule for the mismatch operator is defined accordingly. Since distinct names are different constant names, the condition $a{\neq}b$ is equivalent to $\top$. The transition $[x{\neq}c]\overline{a}b.T \xrightarrow{\overline{a}b} T$ for instance is not admitted since the name variable x needs to be instantiated before the mismatch can be evaluated. One advantage of this semantics is the validity of the following lemma.

Lemma 1. *The following statements are valid whenever* $S \xrightarrow{\lambda} T$.

1. $S\alpha \xrightarrow{\lambda\alpha} T\alpha$ *for every renaming* α.
2. $S\sigma \xrightarrow{\lambda} T\sigma$ *for every substitution* σ.
3. $S\rho \xrightarrow{\lambda} T\rho$ *for every assignment* ρ.

The operational semantics of a process calculus draws a sharp line between internal actions (τ-actions) and external actions (non-τ-actions). From the point of view of interaction, the former is unobservable and the latter is observable. A *complete internal action sequence* of a process P is either an infinite τ-action sequence

$$P \xrightarrow{\tau} P_1 \xrightarrow{\tau} \ldots \xrightarrow{\tau} P_i \xrightarrow{\tau} \ldots$$

or a finite τ-action sequence $P \xrightarrow{\tau} P_1 \xrightarrow{\tau} \ldots \xrightarrow{\tau} P_n$, where P_n cannot perform any τ-action. A process P is *divergent* if it has an infinite internal action sequence; it is *terminating* otherwise.

In the π-calculus with the uniform treatment of names, *if_then_else_* command is defined with the help of the unguarded choice operator. A nice thing about the present semantics is that one may define *if_then_else_* command in the following manner.

$$\textit{if } m{=}n \textit{ then } S \textit{ else } T \quad \stackrel{\text{def}}{=} \quad [m{=}n]S \mid [m{\neq}n]T.$$

The idea can be generalized. Suppose $\{\varphi_i\}_{1\leq i\leq n}$ is a finite collection of *disjoint conditions*, meaning that $\models \varphi_i \wedge \varphi_j \Rightarrow \bot$ for all $i, j \leq n$ such that $i \neq j$. The *conditional choice* $\sum_{i\in\{1,..,n\}} \varphi_i T_i$ is defined in the following fashion.

$$\sum_{1\leq i\leq n} \varphi_i T_i \quad \stackrel{\text{def}}{=} \quad \varphi_1 T_1 \mid \ldots \mid \varphi_n T_n. \tag{10}$$

In practice most choice operations are actually conditional choices [18]. Another form of choice is the so-called *internal choice*, defined as follows:

$$\sum_{1\leq i\leq n} \psi_i \tau.T_i \quad \stackrel{\text{def}}{=} \quad (c)(\psi_1 c.T_1 \mid \ldots \mid \psi_n c.T_n \mid \overline{c}), \tag{11}$$

where $\{\psi_i\}_{1\leq i\leq n}$ is a collection of conditions and c appears in none of $T_1, \ldots, T_n$.

In [28] it is shown that the replication, the fixpoint operation and the parametric definition are equivalent in the π-calculus, as long as all the choices are guarded. The fixpoint operator plays an indispensable role in proof systems

for regular processes. The parametric definition is strictly more powerful in some variants of the π-calculus. Proof systems for regular π-processes will not be a major concern of this paper. In all the qualified name-passing calculi studied in the present paper the parametric definition is equivalent to the replicator. We shall therefore ignore both the fixpoint and the parametric definition in the rest of the paper.

Some more notation and terminology need be defined. Given an action λ, we may define $\overline{\lambda}$ as follows:

$$\overline{\lambda} \stackrel{\text{def}}{=} \begin{cases} \overline{a}b, & \text{if } \lambda = ab, \\ ab, & \text{if } \lambda = \overline{a}b, \\ ab, & \text{if } \lambda = \overline{a}(b), \\ \tau, & \text{if } \lambda = \tau. \end{cases}$$

The notation $\overline{\ell}$ should be understood accordingly. We shall write $\widetilde{\ }$ for a finite sequence of something of same kind. For example a finite sequence of the names $c_1, \ldots, c_n$ can be abbreviated to $\widetilde{c}$. Let $\Longrightarrow$ be the reflexive and transitive closure of $\stackrel{\tau}{\longrightarrow}$. We write $\stackrel{\widehat{\lambda}}{\longrightarrow}$ for the syntactic equality $\equiv$ if $\lambda = \tau$ and for $\stackrel{\lambda}{\longrightarrow}$ otherwise. The notation $\stackrel{\widehat{\lambda}}{\Longrightarrow}$ stands for $\Longrightarrow\stackrel{\widehat{\lambda}}{\longrightarrow}\Longrightarrow$. If $\lambda^* = \lambda_1 \ldots \lambda_n$, we write $P \stackrel{\lambda^*}{\Longrightarrow}$ if $P \stackrel{\lambda_1}{\Longrightarrow} P_1 \ldots \stackrel{\lambda_n}{\Longrightarrow} P_n$ for some $P_1, \ldots, P_n$. If $P \stackrel{\lambda^*}{\Longrightarrow} P'$, we say that P' is a *descendant* of P. Notice that P is a descendant of itself.

In sequel a relation always means a binary relation on the processes of the π-calculus or one of its variants. If $\mathcal{R}$ is a relation, $\mathcal{R}^{-1}$ is the reverse of $\mathcal{R}$ and $P\mathcal{R}Q$ for membership assertion. If $\mathcal{R}'$ is another relation, the composition $\mathcal{R};\mathcal{R}'$ is the relation $\{(P, Q) \mid \exists L.P\mathcal{R}L \wedge L\mathcal{R}'Q\}$.

2.3 Variants

A number of 'subcalculi' of π have been studied. These variants are obtained by omitting some operators. They can also be obtained by restricting the use of the received names, or the use of continuation, or the forms of prefix etc.. In this section we give a brief summary of some of the variants. Our definitions of the variants are slightly different from the popular ones, since we attempt to give a systematic classification of the π-variants.

The guarded choice is a useful operator in encoding. It is also a basic operator in proof systems. The independence of the choice operator from the other operators of the π-calculus is established in for example [67, 28]. A lot of programming can be carried out in the π-calculus using prefix terms rather than the guarded choice terms [103, 104]. We will write π^- for the subcalculus of π obtained by replacing the guarded choice terms by the prefix terms of the form $\pi.T$. In many aspects π^- is just as good as π. It is for example complete in the sense that one may embed the recursion theory [78] in π^-. See [27] for detail. The calculus π^- can be further slimmed down by removing the match and the mismatch operators. We shall call it the *minimal π-calculus* and shall denote it by π^M. It is conjectured that π^M is considerably weaker than π^-. The intuition behind the conjecture is that the if-command cannot be faithfully encoded in π^M.

Some of the syntactical simplifications of the π-calculus have been studied in literature. Here are some of them:

- Merro and Sangiorgi's local π-calculus forbids the use of a received name as an input channel [52, 53]. The word 'local' refers to the fact that a receiving party may only use a received local name to call upon subroutines delivered by the sending party. The calculus can be seen as a theoretical foundation of the concurrent and distributed languages like *Pict* [74] and *Join* [22]. The local variants we introduce in this paper are obtained from the π-calculus by restricting the use of the received names. A received name may be used as an output channel or an input channel. It may also appear in the object position. This suggests the following three variants. In π^L the grammar for prefix is

  $$\pi \quad := \quad a(x) \mid \overline{n}m.$$

  In π^R the syntax of prefix is

  $$\pi \quad := \quad n(x) \mid \overline{a}m,$$

  and in π^S it is

  $$\pi \quad := \quad n(x) \mid \overline{n}c.$$

In π^S only control information may be communicated, data information may not be passed around. A piece of control information may be sent once, it may not be resent.

Other well known variants are syntactical simplifications of π^-. Let's see a couple of them.

- The π-process $!a(x).T$ can be seen as a method that can be invoked by a process of the form $\overline{a}c.S$. This object oriented programming style is typical of the name-passing calculi. The object may invoke the method several times. It follows that the method must be perpetually available. The minimal modification of π that embodies this idea is π^O-calculus with following grammar:

$$S, T \quad := \quad \mathbf{0} \mid \overline{n}m.T \mid S \mid T \mid (c)T \mid [p{=}q]T \mid [p{\neq}q]T \mid !n(x).T.$$

  The superscript O refers to the fact that π^O-calculus has only proper output prefix; it also reminds of the object oriented programming style. The dual of π^O-calculus, π^I-calculus, is defined by the following grammar.

$$S, T \quad := \quad \mathbf{0} \mid n(x).T \mid S \mid T \mid (c)T \mid [p{=}q]T \mid [p{\neq}q]T \mid !\overline{n}m.T.$$

  The relationship between π^I and π^O is intriguing. There is a straightforward encoding $[\![\_]\!]^\iota$ from the former to the latter defined as follows:

$$\begin{aligned} [\![n(x).T]\!]^\iota &\stackrel{\text{def}}{=} \overline{n}(c).!c(x).[\![T]\!]^\iota, \\ [\![!\overline{n}m.T]\!]^\iota &\stackrel{\text{def}}{=} !n(x).\overline{x}m.[\![T]\!]^\iota. \end{aligned}$$

  The soundness of this encoding is unknown. Both π^O and π^I have too weak control power to be considered as proper models. In π^O for example, a name cannot be used to input two ordered pieces of information since a sending party would get confused. There is no way to define for example a polyadic prefix term like $a(x,y).T$. We shall not consider these two variants in the rest of the paper.

- Honda and Tokoro's object calculus [41, 42, 43] and Boudol's asynchronous π-calculus [12] are based on the idea that an output prefix with a continuation does not interact with any environment. Instead it evolves into a process representing the continuation and an atomic output process whose sole function is to send a name to an environment. This is captured by the following transitions.

$$\overline{a}c.T \quad \stackrel{\tau}{\longrightarrow} \quad \overline{a}c \mid T, \tag{12}$$

$$\overline{a}c \mid a(x).T \quad \stackrel{\tau}{\longrightarrow} \quad \mathbf{0} \mid T\{c/x\}. \tag{13}$$

  (13) is absolutely necessary, whereas (12) can be avoided in view of the fact that $\overline{a}c.T$ must be equal to $\overline{a}c \mid T$. Moreover if we think asynchronously the replication, match and mismatch operators should not apply to the processes of the form $\overline{n}m$. This explains the following grammar of π^A:

$$S, T \quad := \quad \mathbf{0} \mid \overline{n}m \mid \sum_{i \in I} \psi_i n_i(x).T \mid S \mid T \mid (c)T \mid !n(x).T,$$

  where ψ is a finite sequence of match/mismatch operations. The standard prefix operators can be mimicked in π^A in the following way [12]:

$$[\![p(x).S]\!] \stackrel{\text{def}}{=} p(u).(d)(\overline{u}d \mid d(x).[\![S]\!]), \tag{14}$$

$$[\![\overline{p}q.T]\!] \stackrel{\text{def}}{=} (c)(\overline{p}c \mid c(v).(\overline{v}q \mid [\![T]\!])). \tag{15}$$

  This encoding of the output prefix is however not very robust from the observational point of view. In fact [17] have proved that no encodings of the output prefix exist that preserve must equivalence. If divergence is not seen as an important issue, then there are interesting encodings into the asynchronous π-calculus as shown in [65] and in [63]. The asynchronous calculi have a very different flavor from the synchronous calculi. We will not discuss π^A in the rest of the paper. But see [25] for an alternative approach to the asynchronous process calculi.

- A more distant relative is Sangiorgi's *private π-calculus* [87], initially called πI-calculus [82]. The grammar of π^P differs from that of π in the definition of prefix. In π^P it is given by the following BNF.

$$\pi \quad := \quad n(x) \mid \overline{n}(c).$$

  Unlike all the above variants, the π^P-calculus has a different action set than the π-calculus. There are no free output actions. One has typically that

$$a(x).S \mid \overline{a}(c).T \xrightarrow{\tau} (c)(S\{c/x\} \mid T).$$

  In π^P the name emitted by an output action is always local. It is worth remarking that this particular π^P-calculus is not equivalent to the version in which replications are abolished in favor of parametric definitions [82, 28]. It is interesting to compare π^P-calculus with π^S-calculus. In both models the messages communicated at run time are foreordained at compile time. The difference is that in π^P-calculus only local messages can be released, whereas in π^S-calculus global messages may also be transmitted. Despite the results established in [8], the variant π^P appears much less expressive than π^S. For one thing the match and mismatch operators in π^P are redundant. So it is more precise to say that π^P is a variant of π^M. Since π^P has a different action set than π^M, we shall not discuss it in this paper.

One may combine the restrictions introduced in the above variants to produce even more restricted 'subcalculi'. Some of these 'subcalculi' can be rejected right away. For example the variant in which the received names can only be used as data is equivalent to CCS. Another counter example is the variant in which both the input capability and the output capability are perpetual. This calculus is not very useful since no computations in this model ever terminate. It simply does not make sense to talk about Turing completeness for this particular calculus. Merro and Sangiorgi's local π-calculus is the asynchronous version of π^L without the match/mismatch operators. In [66, 16] the authors look at some asynchronous versions of π^I and π^O. Their work puts more emphasis on the perpetual availability of the input/output actions.

A crucial question can be asked about each of the variants: which are complete? At the moment we know that $\pi, \pi^L, \pi^R, \pi^S, \pi^M$ are complete. See [27] for a formal definition of completeness and [105] for additional discussions and proofs.

3 Equality

The starting point of observational theory is to do with the definition of process equality. It was realized from the very beginning [54, 84] that the equalities for processes ought to be observational since it is the effects that processes have in environments that really matter. Different application platforms may demand different observing powers, giving rise to different process equalities. This interactive viewpoint [58] lies at the heart of the theory of equality. In a distributed environment for example, a process is subject to perpetual interventions from a potentially unbounded number of observers in an interleaving manner. Here the right equivalences are bisimulation equivalences. In a one shot testing scenario, equivalent processes are indistinguishable by any single tester in an exclusive fashion. It is nobody's business what the equivalent processes would turn into after a test. This is the testing equivalence. In this section we take a look at both the bisimulation approach and the testing approach.

3.1 Absolute Equality

A reasonable criterion for equality of *self evolving* objects is the famous bisimulation property of Milner [56] and Park [69].

Definition 2. *A relation $\mathcal{R}$ is a* weak bisimulation *if the following statements are valid.*

1. *If $Q\mathcal{R}^{-1}P \xrightarrow{\tau} P'$ then $Q \Longrightarrow Q'\mathcal{R}^{-1}P'$ for some Q'.*
2. *If $P\mathcal{R}Q \xrightarrow{\tau} Q'$ then $P \Longrightarrow P'\mathcal{R}Q'$ for some P'.*

The bisimulation property captures the intuition that an equivalence for self evolving objects should be *maintainable* by the equivalent objects themselves when left alone. It is no good if two self evolving objects were equivalent one minute ago and will have to evolve into inequivalent objects one minute later. Bisimulation is about self evolving activities, not about interactive activities. What more can be said about self evolution? An evolution path may experience a series of state changes. At any particular time of the evolution, the process may turn into an equivalent state, and it may also move to an inequivalent state. Inequivalent states generally exert different effects on environments, while equivalent states always have the same interactive behaviors. Definition 2 can be criticized in that it overlooks the important difference between these two kinds of state transitions. The rectification of Definition 2 is achieved in [101] by the branching bisimulation. The following particular formulation is given by Baeten [4].

Definition 3. *A binary relation $\mathcal{R}$ is a* bisimulation *if it validates the following* bisimulation property.

1. *If $Q\mathcal{R}^{-1}P \xrightarrow{\tau} P'$ then one of the following statements is valid.*
   (a) *$Q \Longrightarrow Q'$ for some Q' such that $Q'\mathcal{R}^{-1}P$ and $Q'\mathcal{R}^{-1}P'$.*
   (b) *$Q \Longrightarrow Q''\mathcal{R}^{-1}P$ for some Q'' such that $Q'' \xrightarrow{\tau} Q'\mathcal{R}^{-1}P'$ for some Q'.*
2. *If $P\mathcal{R}Q \xrightarrow{\tau} Q'$ then one of the following statements is valid.*
   (a) *$P \Longrightarrow P'$ for some P' such that $P'\mathcal{R}Q$ and $P'\mathcal{R}Q'$.*
   (b) *$P \Longrightarrow P''\mathcal{R}Q$ for some P'' such that $P'' \xrightarrow{\tau} P'\mathcal{R}Q'$ for some P'.*

Since we take the branching bisimulations as *the* bisimulations, we leave out the adjective.

A process not only evolves on its own, it also interacts with other processes. A plain criterion for the equalities of *interacting* objects is that an equality between two objects should be *maintainable* no matter how environments may interact with them. It is extremely important to get it right what an environment can and cannot do. In a distributed scenario, an environment may interact with a process placed in the regional network. An environment is not capable of grabbing a *running* process and reprogram it as it were. So it would not be appropriate to admit, say $a(x).(\_\{x/a\} \mid O)$, as an environment.

Definition 4. *An* environment *is a process of the form $(\widetilde{c})(\_ \mid O)$ with the specified hole indicated by '$\_$'.*

A prerequisite for two processes P, Q to be observationally equivalent is that no environment can tell them apart. But saying that the environment $(\widetilde{c})(\_ \mid O)$ cannot observe any difference between P and Q is nothing but saying that $(\widetilde{c})(P \mid O)$ and $(\widetilde{c})(Q \mid O)$ are observationally equivalent. Hence the next definition.

Definition 5. *A relation $\mathcal{R}$ is* extensional *if the following statements are valid.*

1. *If $L\mathcal{R}M$ and $P\mathcal{R}Q$ then $(L \mid P)\ \mathcal{R}\ (M \mid Q)$.*
2. *If $P\mathcal{R}Q$ then $(c)P\ \mathcal{R}\ (c)Q$.*

Process equivalences are *observational*. A process P is *observable*, written $P\Downarrow$, if $P \Longrightarrow\!\xrightarrow{\ell} P'$ for some P'. It is *unobservable*, written $P\not\Downarrow$, if it is not observable. Occasionally we write $P\Downarrow_\ell$ for $\exists P'.P \Longrightarrow\!\xrightarrow{\ell} P'$ and similarly $P\downarrow_\ell$ for $\exists P'.P \xrightarrow{\ell} P'$. It should be apparent that two equivalent processes must be both observable or both unobservable.

Definition 6. *A relation $\mathcal{R}$ is* equipollent *if $P\Downarrow \Leftrightarrow Q\Downarrow$ whenever $P\mathcal{R}Q$.*

Equipollence is the weakest condition that an observational equivalence has to satisfy. Now suppose P and Q are observationally inequivalent. If we ignore the issue of divergence, then there must exist some ℓ such that $P\Downarrow_\ell$ and for all ℓ' such that $Q\Downarrow_{\ell'}$, the actions ℓ and ℓ' exert different effects on some environment $(\widetilde{c})(\_ \mid O)$. As we mentioned just now, what this means is that $(\widetilde{c})(P \mid O)$ and $(\widetilde{c})(Q \mid O)$ are observationally inequivalent. So we may repeat the argument for $(\widetilde{c})(P \mid O)$ and $(\widetilde{c})(Q \mid O)$. But if the calculus is strong enough, say it is complete, then the inequivalence eventually boils down to the fact that one delivers a result at some name and the other fails to do so at any name. Notice that the localization operator plays a crucial role in this argument.

The theory of process calculus has been criticized for not paying enough attention to the issue of divergence. This criticism is more or less to the point. If the theory of process calculus is meant to be an extension of the theory of computation, a divergent computation must be treated in a different way than a terminating computation. How should we formulate the requirement that process equivalences should be divergence respecting. A property that is not only divergence preserving but also consistent with the idea of bisimulation is codivergence.

Definition 7. *A* relation *is* codivergent *if $P\mathcal{R}Q$ implies the following property.*

1. *If $Q \xrightarrow{\tau} Q_1 \xrightarrow{\tau} \cdots \xrightarrow{\tau} Q_n \xrightarrow{\tau} \cdots$ is an infinite internal action sequence, then there must be some $k \geq 1$ and P' such that $P \stackrel{\tau}{\Longrightarrow} P' \mathcal{R} Q_k$.*
2. *If $P \xrightarrow{\tau} P_1 \xrightarrow{\tau} \cdots \xrightarrow{\tau} P_n \xrightarrow{\tau} \cdots$ is an infinite internal action sequence, then there must be some $k \geq 1$ and Q' such that $Q \stackrel{\tau}{\Longrightarrow} Q' \mathcal{R}^{-1} P_k$.*

Using the codivergence condition, one may define an equivalence that is advocated in [27] as *the* equality for process calculi.

Definition 8. *The* absolute equality $=$ *is the largest relation such that the following statements are valid.*

1. *It is reflexive.*
2. *It is equipollent, extensional, codivergent and bisimilar.*

Alternatively we could define the absolute equality as the largest equipollent codivergent bisimulation that is closed under the environments.

The virtue of Definition 8 is that it is completely model independent, as long as we take the view that the composition operator and the localization operator are present in all process calculi. From the point of interaction, there cannot be any argument against equipollence and extensionality. From the point of view of computation, there is no question about codivergence and bisimulation. The only doubt one may raise is if Definition 8 is strong enough. We shall demonstrate in this paper that as far as the π-calculus is concerned, the absolute equality is the most appropriate equivalence relation.

This is the right place to state an extremely useful technical lemma [23], the Bisimulation Lemma. Although worded for $=$, Bisimulation Lemma, called X-property by De Nicola, Montanari and Vaandrager [20], is actually valid for all the *observational* equivalences.

Lemma 9. *If $P \Longrightarrow= Q$ and $Q \Longrightarrow= P$, then $P = Q$.*

A distinguished property about the absolute equality is stated in the next lemma. It is discovered by van Glabbeek and Weijland [101] for the branching bisimilarity.

Lemma 10. *If $P_0 \xrightarrow{\tau} P_1 \xrightarrow{\tau} \ldots \xrightarrow{\tau} P_n = P_0$ then $P_0 = P_1 = \ldots = P_n$.*

A consequence of Lemma 10 is that if $P = Q \xrightarrow{\lambda} Q'$, then any simulation $P \xrightarrow{\tau} P_1 \xrightarrow{\tau} \ldots \xrightarrow{\tau} P_n \xrightarrow{\lambda} P'$ of $Q \xrightarrow{\lambda} Q'$ by P must satisfy the property that $P = P_1 = \ldots = P_n$. This phenomenon motivates the following terminologies.

1. $P \xrightarrow{\tau} P'$ is *deterministic*, notation $P \rightarrow P'$, if $P = P'$.
2. $P \xrightarrow{\tau} P'$ is *nondeterministic*, notation $P \xrightarrow{\iota} P'$, if $P \neq P'$.

A deterministic internal action can be ignored. A nondeterministic internal action has to be properly simulated. The notation $\rightarrow^*$ stands for the reflexive and transitive closure of $\rightarrow$ and $\rightarrow^+$ for the transitive closure. We will write $P \nrightarrow$ to indicate that $P \rightarrow P'$ for no P'.

Since the absolute equality applies not just to π and its variants but to all process calculi, the notation '$=$' could be confusing when more than one model is dealt with. To remove the ambiguity, we sometimes write $=^{\mathbb{M}}$ for the absolute equality of model $\mathbb{M}$. The same notation will be applied to other process equivalences. In this paper, we write $\mathbb{V}$ for an arbitrary π-variant.

3.1.1 External Bisimilarity

The external bisimilarity requires that all observable actions are explicitly simulated. This is commonly referred to as branching bisimilarity if the codivergence is dropped [101].

Definition 11. *A codivergent bisimulation of $\mathbb{V}$ is a* $\mathbb{V}$-bisimulation *if the following statements are valid for every $\ell \in \mathcal{L}$.*

1. *If $Q\mathcal{R}^{-1}P \xrightarrow{\ell} P'$ then $Q \Longrightarrow Q'' \xrightarrow{\ell} Q'\mathcal{R}^{-1}P'$ and $P\mathcal{R}Q''$ for some Q', Q''.*
2. *If $P\mathcal{R}Q \xrightarrow{\ell} Q'$ then $P \Longrightarrow P'' \xrightarrow{\ell} P'\mathcal{R}Q'$ and $P''\mathcal{R}Q$ for some P', P''.*

The $\mathbb{V}$-bisimilarity $\simeq_{\mathbb{V}}$ *is the largest* $\mathbb{V}$*-bisimulation.*

$\mathbb{V}$-bisimilarity is a more effective counterpart of the absolute equality $=_{\mathbb{V}}$. The latter provides the intuition, while the former offers a tool for establishing equational properties. The proof of following fact is a standard textbook application of the bisimulation argument.

Lemma 12. *The equivalence $\simeq_{\mathbb{V}}$ is closed under input choice, output choice, composition, localization, match, mismatch, and guarded replication.*

An immediate consequence of Lemma 12 is the inclusion described in the following lemma.

Lemma 13. $\simeq_{\mathbb{V}} \subseteq =_{\mathbb{V}}$.

If the processes of a model has strong enough observing power, then the absolute equality is as strong as the external bisimilarity. This is the case for the π-calculus. The proof of the following theorem resembles a proof in [24].

Theorem 14. *The π-bisimilarity $\simeq_\pi$ coincides with the absolute equality $=_\pi$.*

Proof. In view of Lemma 13, we only have to prove $=_\pi \subseteq \simeq_\pi$. Let $\mathcal{R}$ be the relation

$$\left\{ (P,Q) \;\middle|\; \begin{array}{l} (c_1,\ldots,c_n)(\overline{a_1}c_1 \mid \ldots \mid \overline{a_n}c_n \mid P) =_\pi \\ (c_1,\ldots,c_n)(\overline{a_1}c_1 \mid \ldots \mid \overline{a_n}c_n \mid Q), \\ \{a_1,\ldots,a_n\} \cap gn(P \mid Q) = \emptyset,\ n \geq 0 \end{array} \right\}.$$

To appreciate the relation notice that $(c'c'')(\overline{a'}c' \mid \overline{a''}c'' \mid P) \neq_\pi (c)(\overline{a'}c \mid \overline{a''}c \mid Q)$ for all P, Q such that $\{a', a''\} \cap gn(P \mid Q) = \emptyset$. We prove that $\mathcal{R}$ is an external bisimulation up to $\sim$, where $\sim$ is defined in Section 3.1.2. Now suppose $A =_\pi B$ where

$$\begin{array}{rcl} A & \stackrel{\text{def}}{=} & (c_1,\ldots,c_n)(\overline{a_1}c_1 \mid \ldots \mid \overline{a_n}c_n \mid P), \\ B & \stackrel{\text{def}}{=} & (c_1,\ldots,c_n)(\overline{a_1}c_1 \mid \ldots \mid \overline{a_n}c_n \mid Q), \end{array}$$

such that $\{a_1,\ldots,a_n\} \cap gn(P \mid Q) = \emptyset$ and $n \geq 0$. Consider an action $P \xrightarrow{\ell} P'$ of P. There are four cases.

1. $\ell = ab$. Let C be $\overline{a}b + \overline{a}c.d$ for some fresh name c. By equipollence and extensionality, $A \mid C \xrightarrow{\tau} A' \mid \mathbf{0}$ must be matched up by $B \mid C \Longrightarrow B'' \mid C \xrightarrow{\tau} B' \mid \mathbf{0}$ for some B', B''. Since $B'' \mid C \xrightarrow{\tau} B' \mid \mathbf{0}$ must be a change of state, it must be the case that $A \mid C =_\pi B'' \mid C$. It follows easily from Bisimulation Lemma that $B \rightarrow^* B'' \xrightarrow{ab} B' =_\pi A'$. Clearly

$$\begin{array}{rcl} A' & \equiv & (c_1,\ldots,c_n)(\overline{a_1}c_1 \mid \ldots \mid \overline{a_n}c_n \mid P'), \\ B' & \equiv & (c_1,\ldots,c_n)(\overline{a_1}c_1 \mid \ldots \mid \overline{a_n}c_n \mid Q'), \\ B'' & \equiv & (c_1,\ldots,c_n)(\overline{a_1}c_1 \mid \ldots \mid \overline{a_n}c_n \mid Q''), \end{array}$$

   for some P', Q', Q''. It follows from $B \rightarrow^* B'' \xrightarrow{ab} B'$ that $Q \Longrightarrow Q'' \xrightarrow{ab} Q'$. Moreover $P\mathcal{R}Q''$ and $P'\mathcal{R}Q'$ by definition.

2. $\ell = \overline{a}b$ for some $b \notin \{c_1, \ldots, c_n\}$. Let c, d be fresh and let D be defined by $D \stackrel{\text{def}}{=} a(x).[x{=}b]c + a(x).d$. Now $A \mid D \xrightarrow{\tau} A' \mid [b{=}b]c$ must be simulated by

$$B \mid D \Longrightarrow B'' \mid D \xrightarrow{\tau} B' \mid [b{=}b]c.$$

The rest of the argument is the same as in the previous case.

3. $\ell = \overline{a}(b)$ and $b \notin \{c_1, \ldots, c_n\}$. Let E be the following process

$$a(x).[x \notin gn(P \mid Q)]\overline{a_{n+1}}x + a(x).d,$$

where d, a_{n+1} are fresh. The rest of the argument is similar.

4. $\ell = \overline{a}c_i$ for some $i \in \{1, \ldots, n\}$. Let F be the process

$$a(x).a_i(y).[x{=}y]\overline{a'_i}y + a(x).d,$$

where d, a'_i are fresh. The rest of the argument is again similar.

Additionally we need to consider the external actions of P, Q that communicate through one of the names $c_1, \ldots, c_n$. There are seven cases. In each case we are content with defining an environment that forces external bisimulation.

1. $P \xrightarrow{c_i b} P'$. Let H be the following process

$$a_i(z).(\overline{z}b.\overline{a'_i}z + \overline{z}d)$$

for some fresh d, a'_i.

2. $P \xrightarrow{c_i c_j} P'$ such that $i \neq j$. Let I be the following process

$$a_i(z).a_j(y).(\overline{z}y.(\overline{a'_j}y \mid \overline{a'_i}z) + \overline{z}d)$$

for some fresh d, a'_i, a'_j.

3. $P \xrightarrow{c_i c_i} P'$. Let J be the following process

$$a_i(z).(\overline{z}z.\overline{a'_i}z + \overline{z}d)$$

for some fresh d, a'_i.

4. $P \xrightarrow{\overline{c_i}b} P'$. Let K be the following process

$$a_i(z).(z(x).[x{=}b]\overline{a'_i}z + z(x).d)$$

for some fresh d, a'_i.

5. $P \xrightarrow{\overline{c_i}(c)} P'$. Let L be the following process

$$a_i(z).(z(x).[x \notin gn(P \mid Q)](\overline{a_{n+1}}x \mid \overline{a'_i}z) + z(x).d),$$

where d, a_{n+1}, a'_i are fresh.

6. $P \xrightarrow{\overline{c_i}c_j} P'$ such that $i \neq j$. Let M be the following process

$$a_i(z).(z(x).a_j(y).[x{=}y](\overline{a'_j}y \mid \overline{a'_i}z) + z(x).d)$$

for some fresh d, a'_i, a'_j.

π	π^-	π^M	π^L	π^R	π^S
$\surd$	?	?	$\surd$	$\surd$	?

Figure 1: External Characterization of $=_{\mathbb{V}}$.

7. $P \xrightarrow{\overline{c}_i c_i} P'$. Let N be the following process

$$a_i(z).(z(x).[x{=}z]\overline{a}'_i z + z(x).d)$$

for some fresh d, a'_i.

We are done. □

We remark that the output choice is crucial in the above proof, but the input choice is not really necessary. A slightly more complicated proof can be given without using any input choice. At the moment we do not see how the theorem can be established without using the output choice.

The external characterization of the absolute equality for a particular model is an important issue. It has been resolved for π^L, π^R in [105]. It is definitely a technically interesting but probably tricky exercise to remove the remaining question marks in Fig. 1.

Problem 15. *What are the answers to the questions raised in Fig. 1?*

There is a standard way to extend the absolute equality from processes to terms. The following definition is well-known.

Definition 16. *Let $\asymp$ be a process equivalence. Then $S \asymp T$ if $S\rho \asymp T\rho$ for every assignment ρ such that $fv(S \mid T) \subseteq dom(\rho)$.*

The equivalence = on π-terms satisfies two crucial equalities. One is

$$T = [x{=}y]T \mid [x{\neq}y]T. \tag{16}$$

The other is

$$[x{\neq}y]\lambda.T = [x{\neq}y]\lambda.[x{\neq}y]T. \tag{17}$$

Both equalities explain the role of the name variables. It is the viewpoint of this paper that all process equivalences should validate both (16) and (17).

3.1.2 Strong Equality and Weak Equality

The only way to refine the absolute equality in a model independent way is to strengthen the bisimulation property. Among all the refinements of Definition 3, the most well known one is Milner and Park's strong bisimulation [69, 56].

Definition 17. *A symmetric relation $\mathcal{R}$ is a* strong bisimulation *if $P \xrightarrow{\tau} P'\mathcal{R}Q'$ for some P' whenever $P\mathcal{R}Q \xrightarrow{\tau} Q'$.*

It is clear that a strong bisimulation is automatically codivergent. Hence the next definition.

Definition 18. *The* strong equality *$=^s$ is the largest reflexive, equipollent, extensional, strong bisimulation.*

The strong equality is the same as the strong bisimilarity of Milner [56].

Definition 19. *A strong bisimulation is an* external strong bisimulation *if the following statements are valid for every $\ell \in \mathcal{L}$.*

1. *If $Q\mathcal{R}^{-1}P \xrightarrow{\ell} P'$ then $Q \xrightarrow{\ell} Q'\mathcal{R}^{-1}P'$ for some Q'.*

2. *If* $P\mathcal{R}Q \xrightarrow{\ell} Q'$ *then* $P \xrightarrow{\ell} P'\mathcal{R}Q'$ *for some* P'.

The external strong bisimilarity $\sim$ *is the largest external strong bisimulation.*

One finds a very useful application of the strong equality in the 'bisimulation up to $\sim$' technique [85]. The strong equality equates all the structurally equivalent processes. Some well known equalities are stated below.

Lemma 20. *The following equalities are valid.*

1. $P \mid \mathbf{0} \sim P$; $P \mid Q \sim Q \mid P$; $P \mid (Q \mid R) \sim (P \mid Q) \mid R$.
2. $(c)\mathbf{0} \sim \mathbf{0}$; $(c)(d)P \sim (d)(c)P$.
3. $(c)(P \mid Q) \sim P \mid (c)Q$ *if* $c \notin gn(P)$.
4. $[a{=}a]P \sim [a{\neq}b]P \sim P$; $[a{\neq}a]P \sim [a{=}b]P \sim \mathbf{0}$.
5. $!\pi.P \sim \pi.P \mid !\pi.P$.

In the other direction we may weaken the absolute equality by relaxing the bisimulation property *a la* Definition 2.

Definition 21. *The* weak equality $=^w$ *is the largest reflexive, equipollent, extensional, codivergent, weak bisimulation.*

The external characterization of the weak equality is Milner's weak bisimilarity [56] enhanced with the codivergence condition.

Definition 22. *A codivergent bisimulation is an* external weak bisimulation *if the following statements are valid for every* $\ell \in \mathcal{L}$.

1. *If* $Q\mathcal{R}^{-1}P \xrightarrow{\ell} P'$ *then* $Q \overset{\ell}{\Longrightarrow} Q'\mathcal{R}^{-1}P'$ *for some* Q'.
2. *If* $P\mathcal{R}Q \xrightarrow{\ell} Q'$ *then* $P \overset{\ell}{\Longrightarrow} P'\mathcal{R}Q'$ *for some* P'.

The external weak bisimilarity $\simeq$ *is the largest external weak bisimulation.*

The proof of Theorem 14 can be extended to a proof of the following coincidence result.

Proposition 23. *The equivalences* $=^w, \simeq$ *coincide in* π^M, π^- *and* π.

The relationships between the strong equality, the absolute equality and the weak equality are stated in the next proposition.

Proposition 24. *The inclusions* $=^s \subsetneq = \subsetneq =^w$ *are strict.*

Proof. It is well known that $a.(\tau.b+\tau.c) =^w a.(\tau.b+\tau.c)+a.c$ holds but $a.(\tau.b+\tau.c) = a.(\tau.b+\tau.c)+a.c$ is not valid. □

3.1.3 Remark

In the theory of bisimulation semantics, the three major contributions are the bisimulation of Milner [56] and Park [69], the branching bisimulation of van Glabbeek and Weijland [101], and the barbed bisimulation of Milner and Sangiorgi [61]. Despite its nice properties [20, 98, 21, 6], the branching bisimulation is not as widely appreciated as it deserves. The absolute equality is introduced in [27] with three points of view. The first is that bisimulation is a defining property for the internal actions, it is a derived property for the external actions. This is very much the motivation for the barbed bisimulation. The second is that a nondeterministic internal action is very different from a deterministic one. The former must be strongly bisimulated, whereas the latter can be weakly bisimulated. This is in our opinion the philosophy of the branching bisimulation. The third is that codivergence and bisimulation complement each other. Deterministic internal actions are ignorable locally but not globally. The notion of codivergence is formulated by

Priese [75]. The correlation of the codivergence property to the bisimulation property is emphasized independently by Fu [27] and by van Glabbeek, Luttik and Trcka [100].

Several other bisimulation based equivalences have been proposed for the π-calculus. First of all let's take a look at the first equivalence proposed for the π-calculus [60], the early equivalence. For the sake of illustration let's denote by $\dot{\approx}$ the largest equipollent codivergent weak bisimulation. We say that P, Q are early equivalent, notation $P \approx_e Q$, if $(\widetilde{c})(P \mid O) \dot{\approx} (\widetilde{c})(Q \mid O)$ for every environment $(\widetilde{c})(\_ \mid O)$. Despite the result in [79], it is still an open problem if $\approx_e$ coincides with the weak bisimilarity $\simeq$. But the issue is not that important in view of the following incompatibility one finds in the definition of $\approx_e$.

- The bisimulation property together with the equipollence property assume that the environments are dynamically changing.
- The closure under the environments in the beginning of the observation says the opposite.

The so-called late equivalence [60, 72] has also been studied in literature, especially when one constructs proof systems. The late equivalence is not really an observational equivalence. As long as the one step interactions are atomic actions, there is no way to tell apart by a third party the difference between $a(x).P + a(x).Q$ and $a(x).P + a(x).Q + a(x).([x{=}d]P + [x{\neq}d]Q)$. Even if the late equivalence is useful when names are treated uniformly, it is a less attractive equivalence when name dichotomy is applied.

Another well known equivalence for the π-calculus is the open bisimilarity [83]. The open approach was meant to deal with the open π-terms head-on. A closely related equivalence is the quasi open bisimilarity [86, 24]. An open bisimulation is a distinction indexed set of relations. In the presence of the name dichotomy, an indexed family of relations is no longer necessary. The following is the standard definition of the open bisimilarity iterated in the present framework.

A codivergent bisimulation $\mathcal{R}$ on $\mathcal{T}$ is an *open bisimulation* if the following statements are valid.

1. If $S\mathcal{R}T$ then $S\sigma\,\mathcal{R}\,T\sigma$ for every substitution σ.
2. If $T\mathcal{R}^{-1}S \xrightarrow{\ell} S'$ then $T \stackrel{\ell}{\Longrightarrow} T'\mathcal{R}^{-1}S'$ for some T'.
3. If $S\mathcal{R}T \xrightarrow{\ell} T'$ then $S \stackrel{\ell}{\Longrightarrow} S'\mathcal{R}T'$ for some S'.

The *open bisimilarity* $\approx_o$ is the largest open bisimulation.

The relation $\approx_o$ should be given up. For one thing it fails (16). The term $\overline{a}a$ for example cannot be simulated by $[x{=}y]\overline{a}a \mid [x{\neq}y]\overline{a}a$. If one tries to correct the above definition, one soon realizes that what one gets is the weak equality.

3.1.4 Algebraic Property

This section takes a closer look at the absolute equality on the π-terms. The goal is to reduce the proof of an equality between π-terms to the proof of an equality between π-processes. To describe the reductions, we need to fix some terminology and notation. We will write $\subseteq_f$ for the finite subset relationship. If $\mathcal{F} \subseteq_f \mathcal{N} \cup \mathcal{N}_v$, then $\mathcal{F}_c$ stands for $\mathcal{F} \cap \mathcal{N}$ and $\mathcal{F}_v$ for $\mathcal{F} \cap \mathcal{N}_v$. Given such a finite set $\mathcal{F}$, a condition ψ may completely characterize the relationships between the elements of $\mathcal{F}$. This intuition is formalized in the next definition.

Definition 25. *Suppose $\mathcal{F} \subseteq_f \mathcal{N} \cup \mathcal{N}_v$. We say that a satisfiable condition ψ is* complete on $\mathcal{F}$, *if $\mathcal{F}_c = n(\psi)$, $\mathcal{F}_v = v(\psi)$ and for every $x \in \mathcal{F}_v$ and every $q \in \mathcal{F}$ it holds that either $\psi \Rightarrow x = q$ or $\psi \Rightarrow x \neq q$. A condition ψ is* complete *if it is complete on $n(\psi) \cup v(\psi)$.*

The defining property of Definition 25 immediately implies the following lemma.

Lemma 26. *Suppose φ is complete on $\mathcal{F}$ and $n(\psi) \cup v(\psi) \subseteq \mathcal{F}$. Then either $\varphi \Rightarrow \psi$ or $\varphi\psi \Rightarrow \bot$.*

It follows from Lemma 26 that if both φ, ψ are complete on $\mathcal{F}$ then either $\varphi \Leftrightarrow \psi$ or $\varphi\psi \Leftrightarrow \bot$. This fact indicates that we may apply complete conditions to carry out case analysis when reasoning about term equality. A case analysis is based on a complete disjoint partition.

Definition 27. *Suppose $\mathcal{F} \subseteq_f \mathcal{N} \cup \mathcal{N}_v$. A finite set $\{\varphi_i\}_{i\in I}$ is a* complete disjoint partition *of $\mathcal{F}$ if the following conditions are met.*

1. *φ_i is complete on $\mathcal{F}$ for every $i \in I$.*
2. *$\bigvee_{i\in I} \varphi_i \Leftrightarrow \top$.*
3. *$\varphi_i \wedge \varphi_j \Rightarrow \bot$ for all i, j such that $i \neq j$.*

Given a set $\mathcal{F} \subseteq_f \mathcal{N} \cup \mathcal{N}_v$, there is always a complete disjoint partition on $\mathcal{F}$. So one can always apply a partition to a π-term.

Lemma 28. *$T \sim \sum_{i\in I} \varphi_i T$, where $\{\varphi_i\}_{i\in I}$ is a complete disjoint partition of $gn(T) \cup fv(T)$.*

Now suppose φ is complete on $gn(S \mid T) \cup fv(S \mid T)$. Usually it is much easier to prove $\varphi S = \varphi T$ than to prove $S = T$. So there is a strong interest in the result stated next.

Lemma 29. *Let $\{\varphi_i\}_{i\in I}$ be a complete disjoint partition of $gn(S \mid T) \cup fv(S \mid T)$. Then $S = T$ if and only if $\varphi_i S = \varphi_i T$ for every $i \in I$.*

How can we reduce the equality proof of $\varphi S = \varphi T$ further if φ is complete on say $\{a, b, x, y, z\} = gn(S \mid T) \cup fv(S \mid T)$? Suppose φ is $[x{=}y][y{=}a][z{=}b]$. Consider the equality $\varphi S = \varphi T$. This equality is the same as

$$\varphi S\{a/x, a/y, b/z\} = \varphi T\{a/x, a/y, b/z\},$$

which can be reduced to $S\{a/x, a/y, b/z\} = T\{a/x, a/y, b/z\}$. The substitution $\{a/x, a/y, b/z\}$ has the property that it agrees with the condition $x{=}a \wedge y{=}a \wedge z{=}b$ and its domain set and range set are subsets of $\{a, b, x, y, z\}$. Substitutions of this kind are very useful when rewriting π-terms.

Definition 30. *An assignment ρ* agrees with *ψ, and ψ agrees with ρ, if the following statements are valid.*

1. *$v(\psi) \subseteq dom(\rho)$.*
2. *For each $x \in v(\psi)$ and each $a \in n(\psi)$, $\psi \Rightarrow x = a$ if and only if $\rho(x) = a$.*
3. *For all $x, y \in v(\psi)$, $\psi \Rightarrow x = y$ if and only if $\rho(x) = \rho(y)$.*

A substitution σ is induced *by ψ if it agrees with ψ such that $n(\sigma) \subseteq n(\psi)$, $v(\sigma) \subseteq v(\psi)$ and $dom(\sigma) = v(\psi)$.*

There could be many assignments that agree with ψ and many substitutions that are induced by ψ. We shall write ρ_ψ for some assignment that agrees with ψ and σ_ψ for some substitution that is induced by ψ. It should be clear that if ψ is complete, then $\sigma_\psi(x) \neq \sigma_\psi(y)$ if and only if $\psi \Rightarrow x \neq y$ if and only if $\rho_\psi(x) \neq \rho_\psi(y)$. The validity of the next lemma is obvious.

Lemma 31. *$\varphi T \sim \varphi T \sigma_\varphi$.*

So we have reduced the proof of $S = T$ to the proof of $\varphi S\sigma_\varphi = \varphi T\sigma_\varphi$ for some φ complete on $gn(S \mid T) \cup fv(S \mid T)$. Now the match conditions which appear in φ are inessential. We may apply the following lemma to remove all the match conditions.

Lemma 32. *The following statements are valid.*

1. *If $n \notin gn(S \mid T) \cup fv(S \mid T)$, then $[x{\neq}n]S = [x{\neq}n]T$ if and only if $S = T$.*
2. *If $x \notin fv(S \mid T)$, then $[x{=}n]S = [x{=}n]T$ if and only if $S = T$ if and only if $[x{\neq}n]S = [x{\neq}n]T$.*

After Lemma 32 we may assume that an equality to be proved is of the form $\delta_{\mathcal{F}} S = \delta_{\mathcal{F}} T$, where $\mathcal{F} = gn(S \mid T) \cup fv(S \mid T)$ and $\delta_{\mathcal{F}}$ is the conjunction

$$\bigwedge\{x \neq n \mid x \in \mathcal{F}_v,\ n \in \mathcal{F} \text{ and } x, n \text{ are distinct}\}.$$

In the last step of the reduction we apply the following result.

Lemma 33. *$\delta_{\mathcal{F}} S = \delta_{\mathcal{F}} T$ if and only if $S\rho_{\delta_{\mathcal{F}}} = T\rho_{\delta_{\mathcal{F}}}$, where $\mathcal{F} = gn(S \mid T) \cup fv(S \mid T)$.*

It should be remarked that $\delta_{\mathcal{F}} S = \delta_{\mathcal{F}} T$ is an equality between two open π-terms, whereas $S\rho_{\delta_{\mathcal{F}}} = T\rho_{\delta_{\mathcal{F}}}$ is an equality between two π-processes. The proof of Lemma 33 is immediate in view of the following fact.

Lemma 34. *If $P = Q$ then $P\alpha = Q\alpha$ for every renaming α.*

We conclude that all equality proofs for π-terms can be transformed to equality proofs for π-processes.

Lemma 29, Lemma 32, Lemma 33 and Lemma 34 are also valid for $\sim$.

It is difficult to attribute the above lemmas to the original authors. It would be fair to say that the early researchers on the π-calculus were all aware of the properties described in these lemmas. See for example the work of Parrow and Sangiorgi [72] and Lin [49].

3.2 Testing Equivalence

The testing equivalence of De Nicola and Hennessy [19, 32] is probably the best known nonbisimulation equivalence for processes. In the testing approach, one only cares about the *result* of observation, not the *course* of observation. To define a testing equivalence, one needs to explain what the observers are, how the observers carry out tests, and what counts as the result of a test. In the spirit of observation theory, there is not much room for variation. The observers are the environments. A test by the observer $(\widetilde{c})(\_ \mid O)$ on a process P is a *complete* internal action sequence of $(\widetilde{c})(P \mid O)$. The result of a test can be either successful or unsuccessful. From the point of view of observation, the success of a test can only be indicated by an observable action. The only variations that may arise are concerned with the way the successes are reported.

This subsection serves two purposes. One is to demonstrate that the testing theory for the π-calculus is just as simple as that for CCS. The second is to give a model independent characterization of the testing equivalence. In Section 3.2.1 and Section 3.2.3, we will use the *separated choice* to achieve greater observing power. Syntactically the separated choice terms are given by

$$\sum_{i\in I} n_i(x).T_i \text{ and } \sum_{i\in I} \overline{n_i} m_i.T_i.$$

The semantics is defined by the following standard rules:

$$\frac{}{\sum_{i\in I} a_i(x).T_i \xrightarrow{a_i c} T_i\{c/x\}}\ i \in I \qquad \frac{}{\sum_{i\in I} \overline{a_i} c_i.T_i \xrightarrow{\overline{a} c_i} T_i}\ i \in I$$

The separated choice operator allows one to define derived terms like $\tau.S + a(x).T$.

3.2.1 May Equivalence and Must Equivalence

The testing machinery of De Nicola and Hennessy [19] can be summarized as follows.

- Success is indicated by the special action ω.
- In the presence of ω, the observers can be simplified to the environments of the form $\_ \mid O$. There is no need for the localization operator. An observer O is obtained from some process by replacing some occurrences of $\mathbf{0}$ by ω.
- A test of P by O is a complete internal action sequence of $P \mid O$.
- A test of P by O is *DH-successful* if at some state of the test, the ω action is *immediately* fireable. It is unsuccessful otherwise.
- A binary relation $\mathcal{R}$ on processes satisfies the *may predicate* if $P\mathcal{R}Q$ implies that, for every observer O, some test of $P \mid O$ is DH-successful if and only if some test of $Q \mid O$ is DH-successful.
- A binary relation $\mathcal{R}$ on processes satisfies the *must predicate* if $P\mathcal{R}Q$ implies that, for every observer O, all tests of $P \mid O$ are DH-successful if and only if all tests of $Q \mid O$ are DH-successful.

$$
\begin{array}{rcl}
O^{\mathcal{F}}_{\epsilon} & \stackrel{\text{def}}{=} & \tau.\mathbf{0} \\
O^{\mathcal{F}}_{ab,\ell^*} & \stackrel{\text{def}}{=} & \tau.\omega + \overline{a}b.O^{\mathcal{F}}_{\ell^*} \\
O^{\mathcal{F}}_{\overline{a}b,\ell^*} & \stackrel{\text{def}}{=} & \tau.\omega + a(z).([z{\neq}b]\tau.\omega \,|\, [z{=}b]\tau.O^{\mathcal{F}\cup\{b\}}_{\ell^*}) \\
O^{\mathcal{F}}_{\overline{a}(b),\ell^*} & \stackrel{\text{def}}{=} & \tau.\omega + a(z).([z{\in}\mathcal{F}]\tau.\omega \,|\, [z{\notin}\mathcal{F}]\tau.O^{\mathcal{F}\cup\{z\}}_{\ell^*})
\end{array}
$$

Figure 2: Trace Observer.

Using the above terminology, we may define the may equivalence $\approx_{may}$ as the largest relation on the π-processes that satisfies the may predicate, and and the must equivalence $\approx_{must}$, as the largest relation that satisfies the the must predicate. Let's see some illustrating examples.

- $A \,|\, !\tau \approx_{may} A$. The may equivalence ignores divergence.
- $A \,|\, !\tau \not\approx_{must} A$ if A is terminating. The must equivalence is discriminating against divergent processes.
- $A \,|\, !\tau \approx_{must} B \,|\, !\tau$ even if $A \not\approx_{must} B$. It is often said that divergence is catastrophic for must equivalence.

It is obvious from these examples that $\approx_{must}$ is incompatible to both $\approx_{may}$ and Milner's weak bisimilarity, which is really undesirable. The may equivalence behaves better. It is well known that $\approx_{may}$ is nothing but the so called trace equivalence. A proof of this coincidence can be found in [19].

Lemma 35. *$P \approx_{may} Q$ if and only if $\forall \ell^* {\in} \mathcal{L}^*.\ (P \stackrel{\ell^*}{\Longrightarrow}) \Leftrightarrow (Q \stackrel{\ell^*}{\Longrightarrow})$.*

All observational equivalences should subsume the trace equivalence [95]. For the must equivalence this is true only for a set of hereditarily terminating processes. The following definition and lemma are from [19].

Definition 36. *A process is* strongly convergent *if all the descendants of the process are terminating.*

Lemma 37. *$P \approx_{must} Q$ implies $P \approx_{may} Q$ if P, Q are strongly convergent.*

Proof. We start by defining the *trace observer* generated by some $\ell^* \in \mathcal{L}^*$ and some finite subset $\mathcal{F}$ of $\mathcal{N}$. The structural definition is given in Fig. 2. Notice that the input choice and the output choice are necessary to define the trace observer. Without loss of generality assume that $P \not\approx_{may} Q$. Then there must exist some nonempty $\ell^*_1 \in \mathcal{L}^*$ such that $P \stackrel{\ell^*_1}{\Longrightarrow}$ and $\neg(Q \stackrel{\ell^*_1}{\Longrightarrow})$. Let $\mathcal{F}'$ be $gn(P \,|\, Q)$. Obviously $P \,|\, O^{\mathcal{F}'}_{\ell^*_1}$ has an unsuccessful test. Since Q is strongly convergent, it does not induce any infinite computation. It follows that all the tests of $Q \,|\, O^{\mathcal{F}'}_{\ell^*_1}$ are successful. □

3.2.2 Remark

The proceeding section has pinpointed the problems with the must equivalence. Let's summarize.

I. Something must be wrong with the fact that Milner's weak bisimilarity is not a sub-relation of the testing equivalence. Both the weak bisimilarity and the testing equivalence are to blame since neither deals with divergence properly. Moreover there is no excuse for $\approx_{must} \not\subseteq \approx_{may}$.

II. It is a bit odd to introduce a special symbol ω to indicate success. The action ω introduces an asymmetry between testers and testees. In De Nicola and Hennessy's approach, a testing equivalence is always defined for a particular process calculus. The definition may vary from one calculus to another. If the testing equivalence is really a fundamental equivalence for processes, there ought to be a model independent definition of the equivalence that applies to all calculi. The model independent characterization should be able to provide some canonicity for the testing approach.

There have been several attempts [73, 13, 62, 10, 11] to modify the definition of must equivalence in order to resolve issues I and II, Let's review some of the proposals.

I. Brinksma, Rensink and Volger's *should equivalence* [13] and Natarajan and Cleaveland's *fair testing equivalence* [62] are two modifications proposed to address issue I. These two variants are essentially defined over the same success predicate. A test of P by O is *FS-successful* if $Q \stackrel{\omega}{\Longrightarrow}$ whenever $P \mid O \Longrightarrow Q$. A binary relation $\mathcal{R}$ on processes satisfies *fair/should predicate* if $P\mathcal{R}Q$ implies that, for every observer O, all tests of $P \mid O$ are FS-successful if and only if all tests of $Q \mid O$ are FS-successful. Let $\approx_{FS}$ be the largest relation that satisfies this predicate. This equivalence stays between Milner's weak bisimilarity and the trace equivalence.

II. Boreale, De Nicola and Pugliese address issue II in [10, 11]. Instead of using the special symbol ω, they let all the observable actions to indicate success. Their version of the fair/should predicate is based on a slightly different notion of success. A test of P by O is *BDP-successful* at ℓ if $Q \stackrel{\ell}{\Longrightarrow}$ whenever $P \mid O \Longrightarrow Q$. Notice that there is a guarantee at ℓ predicate for every $\ell \in \mathcal{L}$. Boreale, De Nicola and Pugliese's equivalence $\approx_{BDP}$ is the largest reflexive contextual relation satisfying all the guarantee at ℓ predicates. It is proved in [10] that $\approx_{BDP}$ coincides with $\approx_{FS}$ for the calculus considered in [10]. The significance of their approach is that it provides a characterization of the must equivalence without resorting to any testing machinery.

We shall cast more light on $\approx_{FS}$ and $\approx_{BDP}$ next.

3.2.3 Testing Equivalence without Testing

Although the study in [10] is carried out for a particular calculus with a particular notion of context, Boreale, De Nicola and Pugliese's approach to testing equivalence is basically model independent. In this section we shall give an even more abstract characterization of $\approx_{BDP}$ in the style of the absolute equality. We start with a similar abstract characterization of the trace equivalence. The proof of the following lemma is essentially given in [10].

Lemma 38. *The equivalence $\approx_{may}$ is the largest reflexive, equipollent, extensional relation.*

In view of Lemma 38 we may introduce the following definition.

Definition 39. *The* diamond equality $=_\diamond$ *is the largest reflexive, equipollent, extensional relation.*

The diamond equality is about the existence of a successful testing. A logical dual would be about the inevitability of successful testings. For the purpose of introducing such a dual, one needs to strengthen the equipollence condition.

Definition 40. *A process P is* strongly observable, *notation $P\Downarrow\!\!\downarrow$, if $P'\Downarrow$ for all P' such that $P \Longrightarrow P'$. A relation $\mathcal{R}$ is* strongly equipollent *if $P\mathcal{R}Q \Rightarrow (P\Downarrow\!\!\downarrow \Leftrightarrow Q\Downarrow\!\!\downarrow)$.*

After Definition 40, the following definition must be expected.

Definition 41. *The* box equality $=_\Box$ *is the largest reflexive, strongly equipollent, extensional relation.*

Without further ado, we summarize the properties of $=_\Box$ by the next theorem. It says that the box equality improves upon the testing equivalence as expected.

Theorem 42. *The following statements are valid.*

1. *The strict inclusions $= \subsetneq =_\Box \subsetneq =_\diamond$ hold.*
2. *The strict inclusion $=_\Box \subsetneq \approx_{must}$ holds for the strongly convergent π-processes.*
3. *The coincidence of $=_\Box$ and $\approx_{must}$ holds for the finite π-processes.*

Proof. (1) The inclusion $= \subseteq =_\Box$ is valid by definition. The strictness of the inclusion is witnessed by the pair $a.b + a.c$, $a.(\tau.b + \tau.c)$. The proof of Lemma 37 can be reiterated to show $=_\Box \subsetneq =_\diamond$. The strictness is witnessed by the process pair $a.(!b \mid !c)$ and $a.b.(!b \mid !c) + a.c.(!b \mid !c)$.

(2) Suppose $P \approx_\Box Q$ and $P \not\approx_{must} Q$ for strongly convergent processes P, Q. Assume without loss of generality that some observer O exists such that $P \mid O$ has an unsuccessful test while all the tests of $Q \mid O$ are successful. There are two cases:

- Suppose the failure test of $P \mid O$ is the infinite tau action sequence

$$P \mid O \xrightarrow{\tau} (\widetilde{c_1})(P_1 \mid O_1) \ldots \xrightarrow{\tau} (\widetilde{c_i})(P_i \mid O_i) \ldots . \tag{18}$$

  If O performs an infinite *consecutive* tau action sequence in (18), then it must be the case that P has performed only a finite number of interactions with O. According to (1), Q must be able to perform the same finite sequence of external actions. But then $Q \mid O$ has an infinite unsuccessful test of the shape (18), which contradicts the assumption. If O does not perform any infinite *consecutive* tau action sequence in (18), then the strong convergence property guarantees that P must perform an infinite number of external actions in (18). It follows from the assumption $P \not\approx_{must} Q$ that Q cannot perform the same infinite number of external actions. It follows from the finite branching property that $P \neq_{\diamond} Q$ must hold, contradicting the other assumption.

- Suppose the failure test of $P \mid O$ is a finite internal action sequence. Let λ be a fresh label and $\widetilde{c} = gn(P \mid Q \mid O)$. It is clear that $(\widetilde{c})(Q \mid O\{\lambda/\omega\}) \Downarrow\Downarrow$ but not $(\widetilde{c})(P \mid O\{\lambda/\omega\}) \Downarrow\Downarrow$. This is again a contradiction.

We conclude that $=_{\Box} \subseteq \approx_{must}$ holds for the strongly convergent π-processes. The strictness of this inclusion is interesting. Rensink and Vogler [77] point out that

$$a.\mu X.(a.X + b) + a.\mu X.(a.X + c) \tag{19}$$

and

$$a.\mu X.(a.(a.X + c) + b) + a.\mu X.(a.(a.X + b) + c), \tag{20}$$

originally from [7], are testing equivalent. They are not fair/should equivalent. In the terminology of this paper (19) and (20) are not box equal.

(3) We now prove that $P \approx_{must} Q$ implies $P =_{\Box} Q$ for all finite π-processes P, Q. Suppose $P \approx_{must} Q$ and $P \neq_{\Box} Q$ for finite P and Q. Take $\mathcal{F} = gn(P \mid Q)$. Without loss of generality, assume that there were $\widetilde{c}$ and O such that $(\widetilde{c})(P \mid O)\Downarrow\Downarrow$ holds but $(\widetilde{c})(Q \mid O)\Downarrow\Downarrow$ is not valid. Moreover since P, Q are finite, we may assume that O is finite. We may even assume that O does not contain the composition operator. Let $(\widetilde{c'})(Q' \mid O')$ be a descendant of $(\widetilde{c})(Q \mid O)$ such that the followings hold:

- $(\widetilde{c})(Q \mid O) \Longrightarrow (\widetilde{c'})(Q' \mid O') \not\Downarrow$ and $(\widetilde{c'})(Q' \mid O')$ may not perform any τ-actions.

- There is a sequence ℓ^* such that $Q \stackrel{\ell^*}{\Longrightarrow} \xrightarrow{\lambda} \Longrightarrow Q'$ and $O \stackrel{\overline{\ell^*}}{\Longrightarrow} \xrightarrow{\overline{\lambda}} O_1 \Longrightarrow O'$.

Now construct a new process $[\![O]\!]$ from O. The effect of the function $[\![\_]\!]$ on the guarded choice terms can be described as follows:

$$\begin{aligned} [\![a(x).S + a(x).T]\!] &\stackrel{\text{def}}{=} \tau.\omega + a(x).[\![S]\!] + a(x).[\![T]\!], \\ [\![\overline{a}m.S + \overline{a}n.T]\!] &\stackrel{\text{def}}{=} \tau.\omega + \overline{a}m.[\![S]\!] + \overline{a}m.[\![T]\!]. \end{aligned}$$

The function $[\![\_]\!]$ is structural on the other operators. The process $[\![O]\!]$ is further modified as follows: An observer $O^{\omega}_{\ell^*\lambda}$ is obtained from $[\![O]\!]$ by removing all the components $\tau.\omega$ from the choice operations of $[\![O]\!]$ that lie in the path, as it were, beginning at O_1 and ending at O'. It is clear that all tests of $P \mid O^{\omega}_{\ell^*\lambda}$ are successful, whereas there is at least one unsuccessful test of $Q \mid O^{\omega}_{\ell^*\lambda}$. This contradicts to the assumption $P \approx_{must} Q$. □

The counter example given by the processes (19) and (20) point out that the box equality is fairer than the testing equivalence. If we assume that the nondeterminism of the choice operator is implemented in a fair or probabilistic manner, a test of the former by $\mu X.(\overline{a}.X + \overline{b}.\omega)$ may or may not succeed. But all tests of (20) by the tester are successful. In the opinion of Rensink and Vogler [77], the fairness assumption is built in the box equality.

So far all the results in testing theory are model dependent. For example the coincidence between $\approx_{FS}$ and $\approx_{BDP}$ cannot be established in a model independent way. This is because to make sense of the fair/should testing one has to incorporate the special symbol ω into a calculus, which is impossible without knowing the details of the model. Similarly the coincidence of $\approx_{FS}$, $\approx_{BDP}$, $\approx_{must}$ on the strongly convergent processes is also model specific. For a

particular model like the π-calculus one may show that $\approx_{BDP}$ is the largest reflexive, strongly equipollent, extensional relation. But there is no way to prove the validity of this characterization for all models. Similarly the coincidence between $\approx_{BDP}$ and $=_{\Box}$ can only be proved for individual models. The same remark applies to the trace equivalence. There are labeled transition systems for which one might not like to talk about traces at all. An example is a labeled transition semantics for a higher order calculus that defines transitions from processes to abstractions or concretions.

It is in the light of the above discussion that the significance of the diamond equality and the box equality emerge. The former provides *the* universal definition of the trace equivalence, whereas the latter offers *the* universal definition of the testing equivalence. These definitions apply to all models of interaction. For a particular model it is possible to give an explicit counterpart of $=_{\Box}$ or $=_{\Diamond}$. The explicit versions of $=_{\Diamond}$ and $=_{\Box}$ may depend heavily on the details of the model. The relationship between $=_{\Box}$ and its explicit characterization is like that between the absolute equality and the external bisimilarity.

The outline for a model independent testing theory is now complete.

3.3 Remark

The model independent methodology can be applied to analyze dozens of process equivalences defined in literature. In particular it can be applied to evaluate the equivalence relations for example in the linear time branching time spectrum [95]. We have confirmed in this paper that the diamond equality is the right generalization of the trace equivalence, the bottom of the spectrum. On the top of the spectrum, the absolute equality and the weak equality enjoy the model independent characterizations. It would be interesting to take a look at the other equivalences of the spectrum from this angle. Some of the equivalences are studied in the strong case. Their generalizations to the weak case might not be closed under composition. The model independent approach may help us in searching for the correct definitions. Good process equivalences are inedpendent of any particular model.

It must be pointed out that it is often difficult to come up with an external characterization of an equality defined in a model independent manner. Let's take a look at an interesting example. An explicit characterization of the absolute equality for π^A-calculus is tricky. It is definitely different from the popular equivalence widely accepted for π^A. The asynchronous equivalence studied by Honda and Tokoro [41], Boudol [12] and Amadio, Castellani and Sangiorgi [2], satisfies the following equality, where $\simeq_a$ stands for the asynchronous bisimilarity.

$$a(x).\overline{a}x \simeq_a \mathbf{0}. \tag{21}$$

The equality (21) does not hold for the absolute equality since it violates the equipollence condition. Another equality validated by the asynchronous equivalence is (22).

$$!a(x).\overline{a}x \simeq_a a(x).\overline{a}x \tag{22}$$

Equality (22) is rejected by the absolute equality because $\overline{a}c \,|\, !a(x).\overline{a}x$ is divergent but $\overline{a}c \,|\, a(x).\overline{a}x$ is not. However the following absolute equality holds.

$$a(x).\overline{a}x \,|\, a(x).\overline{a}x =_{\pi^A} a(x).\overline{a}x \tag{23}$$

How should we reconcile the difference between the absolute equality and the asynchronous equivalence? A new approach to the asynchronous calculus that gives a satisfactory answer to the question is outlined in [25]. Asynchrony is more of an application issue than a model theoretical issue.

An important topic that is not discussed in the present paper is the pre-order relations on processes. A pre-order can be interpreted as an implementation relation or a refinement relation. Examples of pre-orders are simulation and testing order. Notice that the box equality immediately suggests the box order $\leq_{\Box}$ relation, which is the model independent counterpart of the testing order. So far the order theory of processes has been basically model specific. It should be fruitful to carry out a systematic model independent study on the order relations on the processes. Initial efforts have been made in [31] along this line of investigation.

We conclude this section by remarking that the branching style bisimulation property is what makes the coincidence proofs difficult. External characterizations of the weak equalities of the π-variants are much easier to come by.

4 Expressiveness

How do different variants of the π-calculus compare? One fundamental criterion for comparison is relative expressiveness. Typical quesitons about expressiveness can be addressed in the following fashion: Is a calculus $\mathbb{M}$ as powerful as another calculus $\mathbb{L}$? Or are $\mathbb{L}$ and $\mathbb{M}$ non-comparable in terms of expressiveness? To answer these questions, we need a notion of relative expressiveness that does not refer to any particular model. The philosophy developed in [27] is that the relative expressiveness is the same thing as the process equality. The latter is a relation on one calculus, whereas the former is a relation from one calculus to another. To say that $\mathbb{M}$ is at least as expressive as $\mathbb{L}$ means that for every process L in $\mathbb{L}$ there is some process M in $\mathbb{M}$ such that M is somehow equal to L. It is clear from this statement that relative expressiveness must break the symmetry of the absolute equality. Condition (2) of Definition 8 can be safely kept when comparing two process calculi. But condition (1) has to be modified. The reader is advised to consult [27] for the argument why the reflexivity condition for the absolute equality turns into a totality condition and a soundness condition. The following definition is taken from [27].

Definition 43. *Suppose $\mathbb{L}, \mathbb{M}$ are two process calculi. A binary relation $\mathfrak{R}$ from the set $\mathcal{P}_{\mathbb{L}}$ of $\mathbb{L}$-processes to the set $\mathcal{P}_{\mathbb{M}}$ of $\mathbb{M}$-processes is a* subbisimilarity *if the following statements are valid.*

1. *$\mathfrak{R}$ is* reflexive from $\mathbb{L}$ to $\mathbb{M}$ *in the sense that the following properties hold.*
   - *(a) $\mathfrak{R}$ is* total, *meaning that $\forall L \in \mathcal{P}_{\mathbb{L}}.\exists M \in \mathcal{P}_{\mathbb{M}}.L\mathfrak{R}M$.*
   - *(b) $\mathfrak{R}$ is* sound, *meaning that $M_1\mathfrak{R}^{-1}L_1 =_{\mathbb{L}} L_2\mathfrak{R}M_2$ implies $M_1 =_{\mathbb{M}} M_2$.*
2. *$\mathfrak{R}$ is equipollent, extensional, codivergent and bisimilar.*

We say that $\mathbb{L}$ is subbisimilar to $\mathbb{M}$, *notation $\mathbb{L} \sqsubseteq \mathbb{M}$, if there is a subbisimilarity from $\mathbb{L}$ to $\mathbb{M}$.*

An elementary requirement for the relative expressiveness relationship is transitivity. The subbisimilarity relationship is well defined in this aspect since its transitivity is apparent from the definition. So we have formalized the intuitive notion "$\mathbb{M}$ being at least as expressive as $\mathbb{L}$" by "$\mathbb{L} \sqsubseteq \mathbb{M}$". We write $\mathbb{L} \sqsubset \mathbb{M}$ if $\mathbb{L} \sqsubseteq \mathbb{M}$ and $\mathbb{M} \not\sqsubseteq \mathbb{L}$, meaning that $\mathbb{M}$ is strictly more expressive than $\mathbb{L}$. The reader might wonder why the soundness condition of Definition 43 is not strengthened to the full abstraction. The truth is that the soundness condition is equivalent to the full abstraction condition as far as Definition 43 is concerned.

It is shown in [27] that in general there are an infinite number of pairwise incompatible subbisimilarities from $\mathbb{L}$ to $\mathbb{M}$. However for the π-variants studied in [28] there is essentially a unique subbisimilarity between any two of them. It is shown in [27] that a subbisimilarity for these variants is basically a total relation $\mathfrak{R}$ satisfying the following external characterization property:

1. If $Q\mathfrak{R}^{-1}P \xrightarrow{\ell} P'$ then $Q \Longrightarrow Q'' \xrightarrow{\ell} Q'\mathfrak{R}^{-1}P'$ and $P\mathfrak{R}Q''$ for some Q', Q''.
2. If $P\mathfrak{R}Q \xrightarrow{\ell} Q'$ then $P \Longrightarrow P'' \xrightarrow{\ell} P'\mathfrak{R}Q'$ and $P''\mathfrak{R}Q$ for some P', P''.

Now consider the self translation $[\![\_]\!]_{\bowtie}$ from the π-calculus to itself. The nontrivial part of the translation is defined as follows:

$$\begin{aligned} [\![a(x).T]\!]_{\bowtie} &\stackrel{\text{def}}{=} \overline{a}(c).c(x).[\![T]\!]_{\bowtie}, \\ [\![\overline{a}b.T]\!]_{\bowtie} &\stackrel{\text{def}}{=} a(y).(\overline{y}b \mid [\![T]\!]_{\bowtie}). \end{aligned}$$

It is structural on the non-prefix terms. Is $[\![\_]\!]_{\bowtie}; =$ a subbisimilarity? The negative answer is proved in [27]. This is an example of a typical phenomenon. Without the soundness condition, many liberal encodings like the above one would be admitted. In most cases soundness is the only thing to prove when comparing a syntactical subcalculus against a super calculus.

If the external characterizations of the absolute equality in π_1 and π_2 are available, then it is often easy to see if $\pi_1 \sqsubseteq \pi_2$. Otherwise it might turn out difficult to prove $\pi_1 \sqsubseteq \pi_2$. For example so far we have not been able to confirm that $\pi^M \sqsubseteq \pi^- \sqsubseteq \pi$, although the following negative results are easy to derive.

$\sqsubseteq$	π	π^L	π^R	π^S
π	$\surd$	$\times$	$\times$	?
π^L	$\times$	$\surd$	$\times$	?
π^R	$\times$	$\times$	$\surd$	?
π^S	?	?	?	$\surd$

Figure 3: Expressiveness Relationship

Proposition 44. $\pi \not\sqsubseteq \pi^- \not\sqsubseteq \pi^M$.

Proof. Using the same idea from [27] it is routine to show that the π-process $a(x).\overline{b}b + a(x).\overline{c}c$ cannot be defined in π^- and the π^--process $a(x).[x{=}c]\overline{b}b$ cannot be defined in π^M. □

The relative expressiveness between π, π^L, π^R is settled in [105]. Fig. 3 shows that they are pairwise incompatible. The relative expressiveness of π^S is still unknown.

Before ending this section, let's see a positive result. Consider the boolean expressions constructed from the binary relation = and the logical operators $\wedge, \neg$. For example $\neg(\neg(x = a) \wedge \neg(x = b))$ is such an expression, which is normally abbreviated to $x = a \vee x = b$. Let $\pi^\vee$ denote the π-calculus with the match and mismatch operators replaced by the operator $[\varphi](\_)$, where φ is a boolean expression.

Proposition 45. $\pi \sqsubseteq \pi^\vee \sqsubseteq \pi$.

Proof. The external bisimilarity and the absolute equality coincide for $\pi^\vee$. So $\pi \sqsubseteq \pi^\vee$ is obvious. The proof of the converse is equally simple as soon as the encodings of the processes of the form $[\varphi]T$ become clear. Given any φ, one may transform it into a disjunctive normal form $\bigvee_{i\in I} \varphi_i$. By applying the equivalence

$$\bigvee_{i\in I} \varphi_i \Leftrightarrow \bigvee_{i\in I} \varphi_i \wedge (u = v) \vee \bigvee_{i\in I} \varphi_i \wedge (u \neq v),$$

one gets a complete disjoint partition $\bigvee_{j\in J} \psi_j$ of φ on $fv([\varphi]T) \cup gn([\varphi]T)$. It follows that $[\varphi]T = \prod_{j\in J}[\psi_j]T$ holds in $\pi^\vee$. Therefore $[\varphi]T$ can be bisimulated by $\prod_{j\in J}[\psi_j]T$. □

5 Proof System

An important issue for a process calculus is to design algorithms for the decidable fragments of the calculus. The behavior of a finite term can obviously be examined by a terminating procedure [56]. More generally if a term has only a finite number of descendants and is finite branching, then there is an algorithm that generates its transition graph [47], which can be seen as the abstract representation of the term. An equivalence checker for the terms now works on the transitions graphs. To help transform the terms to their underlying transition graphs, a recursive equational rewrite system would be helpful.

In this section we construct two equational systems of the finite π-terms, one for the absolute equality and the other for the box equality. Our prime objective is to demonstrate how the name dichotomy simplifies equational reasoning.

5.1 Normal Form

To construct an equational system, it is always convenient to introduce a combinator that is capable of describing the nondeterministic choices inherent in concurrent systems. This is the *general choice* operator '+'. The semantics of this operator is defined by the following rules.

$$\frac{S \xrightarrow{\lambda} S'}{S + T \xrightarrow{\lambda} S'} \qquad \frac{T \xrightarrow{\lambda} T'}{S + T \xrightarrow{\lambda} T'}$$

The choice operation is both commutative and transitive. We will write $T_1 + \ldots + T_n$ or $\sum_{i\in I} T_i$, called a *summation*, without further comment. Here T_i is a summand. Occasionally this notation is confused with the one defined in (10) or (11). When this happens, the confusions are harmless. Using the unguarded choice operator, one may define for example $[p{\in}\mathcal{F}]T$ by

$$\sum_{m\in\mathcal{F}} [p{=}m]T.$$

In general one could define $(\varphi)T$ for a boolean expression constructed from $=, \wedge, \neg$. It follows that *if φ then S else T* is definable using the general choice operator. The choice operator is a debatable operator since it destroys the congruence property of the absolute equality. In most of the axiomatic treatment of this operator an induced congruence relation is introduced. This is avoided in this paper since we see the choice operator not as a proper process operator but as an auxiliary one used in the proof systems.

One criterion for an equational system is if it allows one to reduce every term to some normal form. The exercise carried out in Section 3.1.4 suggests the following definitions.

Definition 46. *Let $\mathcal{F}$ be $gn(T) \cup fv(T)$. The π-term T is a* normal form *on $\mathcal{F}$ if it is of the form*

$$\sum_{i\in I} \lambda_i.T_i$$

such that for each $i \in I$ one of the followings holds.

1. *If $\lambda_i = \tau$ then T_i is a normal form on $\mathcal{F}$.*
2. *If $\lambda_i = \overline{n}m$ then T_i is a normal form on $\mathcal{F}$.*
3. *If $\lambda_i = \overline{n}(c)$ then $T_i \equiv [c{\notin}\mathcal{F}]T_i^c$ for some normal form T_i^c on $\mathcal{F} \cup \{c\}$.*
4. *If $\lambda_i = n(x)$ then T_i is of the form*
$$[x{\notin}\mathcal{F}]T_i^{\neq} + \sum_{m\in\mathcal{F}} [x{=}m]T_i^m$$
such that $T_i^{\neq}$ is a normal form on $\mathcal{F} \cup \{x\}$ and, for each $m \in \mathcal{F}$, $x \notin fv(T_i^m)$ and T_i^m is a normal form on $\mathcal{F}$.

Definition 47. *T is a* complete normal form *on $\mathcal{F}$ if it is of the form $\delta_{\mathcal{F}}T'$ for some normal form T' such that $gn(T') \cup fv(T') \subseteq \mathcal{F} \subseteq_f \mathcal{N} \cup \mathcal{N}_v$.*

5.2 Axiom for Absolute Equality

The equational system for the absolute equality on the finite π-terms is given in Fig. 4. The axioms C1 and C2 are *computation laws*. Notice that C2 implies the following equality

$$\tau.T = \tau.(T + \tau.T).$$

We write $AS \vdash S = T$ if $S = T$ can be derived from the axioms and the rules in Fig. 4 and the equivalence and congruence rules. Some derived axioms are summarized in the next lemma. The proofs of these derived axioms can be found in for example [29].

Lemma 48. *The following propositions are valid.*

1. $AS \vdash T = T + \varphi T$.
2. $AS \vdash \varphi\pi.T = \varphi\pi.\varphi T$.
3. $AS \vdash (c)[x{\neq}c]T = (c)T$ *and* $(c)[x{=}c]T = \mathbf{0}$.
4. $AS \vdash \varphi T = \varphi T\sigma_\varphi$.

L1	$(c)\mathbf{0}$	$=$	$\mathbf{0}$	
L2	$(c)(d)T$	$=$	$(d)(c)T$	
L3	$(c)\pi.T$	$=$	$\pi.(c)T$	if $c \notin n(\pi)$
L4	$(c)\pi.T$	$=$	$\mathbf{0}$	if $c = subj(\pi)$
L5	$(c)\varphi T$	$=$	$\varphi(c)T$	if $c \notin n(\varphi)$
L6	$(c)[x{=}c]T$	$=$	$\mathbf{0}$	
L7	$(c)(S+T)$	$=$	$(c)S + (c)T$	
M1	$(\top)T$	$=$	T	
M2	$(\bot)T$	$=$	$\mathbf{0}$	
M3	φT	$=$	ψT	if $\varphi \Leftrightarrow \psi$
M4	$[x{=}p]T$	$=$	$[x{=}p]T\{p/x\}$	
M5	$[x{\neq}p]\pi.T$	$=$	$[x{\neq}p]\pi.[x{\neq}p]T$	if $x \notin bv(\pi) \not\ni p$
M6	$\varphi(S+T)$	$=$	$\varphi S + \varphi T$	
S1	$T + \mathbf{0}$	$=$	T	
S2	$S + T$	$=$	$T + S$	
S3	$R + (S+T)$	$=$	$(R+S)+T$	
S4	$T + T$	$=$	T	
S5	T	$=$	$[x{=}p]T + [x{\neq}p]T$	
S6	$n(x).S + n(x).T$	$=$	$n(x).S + n(x).T + n(x).([x{=}p]S + [x{\neq}p]T)$	
C1	$\lambda.\tau.T$	$=$	$\lambda.T$	
C2	$\tau.(S+T)$	$=$	$\tau.(\tau.(S+T)+T)$	

Figure 4: Axioms for Absolute Equality.

5. $AS \vdash \lambda.\varphi\tau.T = \lambda.\varphi T$.

6. $AS \vdash m(x).([x{\in}\mathcal{F}]T' + [x{\notin}\mathcal{F}]\tau.T) + \sum_{i=1}^{k} m(x).([x{\neq}n_i]T'_i + [x{=}n_i]\tau.T) = m(x).([x{\in}\mathcal{F}]T' + [x{\notin}\mathcal{F}]\tau.T) + \sum_{i=1}^{k} m(x).([x{\neq}n_i]T'_i + [x{=}n_i]\tau.T) + m(x).T$, *where* $\mathcal{F} = \{n_1, \ldots, n_k\}$.

Using these derived axioms it is easy to prove the following lemma by structural induction.

Lemma 49. *Suppose T is a finite π-term and φ is complete on a finite set $\mathcal{F} \supseteq gn(T) \cup fv(T)$. Then $AS \vdash \varphi T = \varphi^{=}\delta T'$ for some complete normal form $\delta T'$ on $gn(\delta T') \cup fv(\delta T')$.*

Proof. To start with, $AS \vdash \varphi T = \varphi^{=}\varphi^{\neq}T = \varphi^{=}(\varphi^{\neq}T)\sigma_{\varphi^{=}}$. In the second step observe that within AS we can carry out the reductions, explained in Section 3.1.4, to $(\varphi^{\neq}T)\sigma_{\varphi^{=}}$. Notice that, for each $\mathcal{F} \subseteq_f \mathcal{N} \cup \mathcal{N}_v$, although $\{x{\notin}\mathcal{F}\} \cup \{x{=}m\}_{m\in\mathcal{F}}$ is not a complete disjoint partition of $\mathcal{F} \cup \{x\}$, the set $\{(x{\notin}\mathcal{F}) \wedge \delta_{\mathcal{F}}\} \cup \{(x{=}m) \wedge \delta_{\mathcal{F}}\}_{m\in\mathcal{F}}$ *is* a complete disjoint partition of $\mathcal{F} \cup \{x\}$. □

Suppose we are to prove $AS \vdash \varphi S = \varphi T$ where φ is complete on $gn(S \mid T) \cup fv(S \mid T)$. According to Lemma 49 we only need to prove $AS \vdash \delta S' = \delta T'$ for some complete normal forms $\delta S', \delta T'$. In the light of this fact the next lemma should play a crucial role in the completeness proof.

Lemma 50. *Suppose S, T are normal forms on the finite set $\mathcal{F} \supseteq gn(S \mid T) \cup fv(S \mid T)$. If $\delta_{\mathcal{F}}S = \delta_{\mathcal{F}}T$ then $AS \vdash \delta_{\mathcal{F}}\tau.S = \delta_{\mathcal{F}}\tau.T$.*

Proof. Assume that $S \equiv \sum_{i\in I} \lambda_i.S_i$ and $T \equiv \sum_{j\in J} \lambda_j.T_j$ and that $\delta_{\mathcal{F}}S = \delta_{\mathcal{F}}T$ for some $\mathcal{F} \supseteq gn(S \mid T) \cup fv(S \mid T)$. Let ρ be an assignment that agrees with $\delta_{\mathcal{F}}$. We are going to establish by simultaneous induction on the structure of S, T the properties stated below.

(S) If $T\rho \rightarrow^{+} T'\rho \nrightarrow$ then $AS \vdash \delta_{\mathcal{F}}\tau.T = \delta_{\mathcal{F}}\tau.(T + T') = \delta_{\mathcal{F}}\tau.T'$.

(P) If $\delta_{\mathcal{F}}S = \delta_{\mathcal{F}}T$ then $AS \vdash \delta_{\mathcal{F}}\tau.S = \delta_{\mathcal{F}}\tau.(S + T) = \delta_{\mathcal{F}}\tau.T$.

Suppose $T\rho \to T_1\rho \to \ldots \to T_n\rho \to T'\rho \not\to$. Lemma 33 and Lemma 34 imply that $\delta_{\mathcal{F}} T_1 = \ldots = \delta_{\mathcal{F}} T_n = \delta_{\mathcal{F}} T'$. By induction hypothesis on (P), $AS \vdash \delta_{\mathcal{F}}\tau.T_1 = \ldots = \delta_{\mathcal{F}}\tau.T_n = \delta_{\mathcal{F}}\tau.T'$. Therefore

$$AS \vdash \delta_{\mathcal{F}}\tau.T = \delta_{\mathcal{F}}\tau.(T + \tau.T').$$

Let $\lambda_j.T_j$ be a summand of T. Consider how $T'\rho$ might bisimulate $T\rho \xrightarrow{\lambda_j} T_j\rho$. There are four cases.

1. $\lambda_j = \tau$ and $T\rho \xrightarrow{\tau} T_j\rho$. Suppose the τ-action is bisimulated by $T'\rho \xrightarrow{\tau} T'_{j'}\rho = T_j\rho$ for some $T'_{j'}$. Then $\delta_{\mathcal{F}} T'_{j'} = \delta_{\mathcal{F}} T_j$ by Lemma 33 and Lemma 34. It follows from the induction hypothesis on (P) that $AS \vdash \delta_{\mathcal{F}}\tau.T'_{j'} = \delta_{\mathcal{F}}\tau.T_j$. Using (2) of Lemma 48 one gets the following inference

$$\begin{array}{rcl} AS \vdash \delta_{\mathcal{F}}\tau.(T + \tau.T') & = & \delta_{\mathcal{F}}\tau.(T + \tau.(T' + \tau.T'_{j'})) \\ & = & \delta_{\mathcal{F}}\tau.(T + \tau.(T' + \tau.T_j)). \end{array}$$

   If $T\rho \xrightarrow{\tau} T_j\rho$ is bisimulated vacuously by $T'\rho$. Then $\delta_{\mathcal{F}} T_j = \delta_{\mathcal{F}} T'$ according to Lemma 33 and Lemma 34 and consequently $AS \vdash \delta_{\mathcal{F}}\tau.T_j = \delta_{\mathcal{F}}\tau.T'$ by the induction hypothesis on (P). It follows from C2 that

$$\begin{array}{rcl} AS \vdash \delta_{\mathcal{F}}\tau.(T + \tau.T') & = & \delta_{\mathcal{F}}\tau.(T + \tau.(T' + \tau.T')) \\ & = & \delta_{\mathcal{F}}\tau.(T + \tau.(T' + \tau.T_j)). \end{array}$$

2. $\lambda_j = \overline{n}m$ and $T\rho \xrightarrow{\overline{a}b} T_j\rho$ for some a, b. We can prove as in the next case that $AS \vdash \delta_{\mathcal{F}}\tau.(T + \tau.T') = \delta_{\mathcal{F}}\tau.(T + \tau.(T' + \overline{n}m.T_j))$.

3. $\lambda_j = \overline{m}(c)$ and $T\rho \xrightarrow{\overline{a}(c)} T_j\rho$ for some a. Suppose this action is bisimulated by $T'\rho \xrightarrow{\overline{a}(c)} T'_{j'}\rho = T_j\rho$ for some $T'_{j'}$. Then $\delta_{\mathcal{F}}[c\notin\mathcal{F}]T'_{j'} = \delta_{\mathcal{F}}[c\notin\mathcal{F}]T_j$ and

$$AS \vdash \delta_{\mathcal{F}}[c\notin\mathcal{F}]\tau.T'_{j'} = \delta_{\mathcal{F}}[c\notin\mathcal{F}]\tau.T_j$$

   by induction hypothesis on (P). Therefore

$$AS \vdash \delta_{\mathcal{F}}\overline{m}(c).[c\notin\mathcal{F}]T'_{j'} = \delta_{\mathcal{F}}\overline{m}(c).[c\notin\mathcal{F}]T_j.$$

   It follows from (3) of Lemma 48 that

$$AS \vdash \delta_{\mathcal{F}}\tau.(T + \tau.T') = \delta_{\mathcal{F}}\tau.(T + \tau.(T' + \overline{m}(c).T_j)).$$

4. $\lambda_j = m(x)$. Assume that $\rho(m) = a$ and $gn((S \mid T)\rho) = \{b_1, \ldots, b_n\}$. Let $n_k \in \mathcal{F}$ be such that $n_k = b_k$ if $b_k \in \mathcal{F}$ and $\rho(n_k) = b_k$ if $b_k \notin \mathcal{F}$. There are two subcases.

   (a) For every $k \in \{1, \ldots, n\}$, one has $T\rho \xrightarrow{ab_k} T_j\rho\{b_k/x\}$. Let this action be bisimulated by $T'\rho \xrightarrow{ab_k} T'_{n_k}\rho\{b_k/x\} = T_j\rho\{b_k/x\}$. Then

$$AS \vdash \delta_{\mathcal{F}}[x{=}n_k]\tau.T'_{n_k} = \delta_{\mathcal{F}}[x{=}n_k]\tau.T_j$$

   by induction hypothesis on (P). Consequently

$$\begin{array}{rcl} \delta_{\mathcal{F}}\tau.(T + \tau.T') & = & \delta_{\mathcal{F}}\tau.(T + \tau.(T' + m(x).T'_{n_k})) \\ & = & \delta_{\mathcal{F}}\tau.(T + \tau.(T' + m(x).([x{\neq}n_k]\tau.T'_{n_k} + [x{=}n_k]\tau.T'_{n_k}))) \\ & = & \delta_{\mathcal{F}}\tau.(T + \tau.(T' + m(x).([x{\neq}n_k]\tau.T'_{n_k} + [x{=}n_k]\tau.T_j))). \end{array}$$

   (b) Suppose $T\rho \xrightarrow{ad} T_j\rho\{d/x\}$, where $d \notin \mathcal{F}$, is simulated by

$$T'\rho \xrightarrow{ad} T_x\rho\{d/x\} = T_j\rho\{d/x\}.$$

   Like in the previous subcase, we may prove that

$$AS \vdash \delta_{\mathcal{F}}\tau.(T + \tau.T') = \delta_{\mathcal{F}}\tau.(T + \tau.(T' + m(x).([x\in\mathcal{F}]\tau.T_x + [x\notin\mathcal{F}]\tau.T_j))).$$

Putting together the two equalities obtained in (a) and (b), we get the following equational rewriting

$$\begin{aligned} AS \vdash \delta_{\mathcal{F}}\tau.(T + \tau.T') &= \delta_{\mathcal{F}}\tau.(T + \tau.(T' + m(x).([x\in\mathcal{F}]\tau.T_x + [x\notin\mathcal{F}]\tau.T_j) \\ &\quad + \sum_{j=1}^{n} m(x).([x\neq n_k]\tau.T_{n_k} + [x=n_k]\tau.T_j))) \\ &= \delta_{\mathcal{F}}\tau.(T + \tau.(T' + m(x).T_j)), \end{aligned}$$

where the second equality holds by (6) of Lemma 48.

Using the fact that $\delta_{\mathcal{F}}T = \delta_{\mathcal{F}}T'$ we may apply the above argument to every summand of T to derive that

$$AS \vdash \delta_{\mathcal{F}}\tau.T = \delta_{\mathcal{F}}\tau.(T + \tau.(T' + T)).$$

Resorting to the full power of C2 we get from the above equality the following:

$$AS \vdash \delta_{\mathcal{F}}\tau.T = \delta_{\mathcal{F}}\tau.(T + T'). \tag{24}$$

If we examine the proof of (24) carefully we realize that it also establishes the following fact:

$$AS \vdash \delta_{\mathcal{F}}\tau.(T + T') = \delta_{\mathcal{F}}\tau.T'.$$

This finishes the inductive proof of (S).

The inductive proof of (P) is now simpler. Suppose $\delta_{\mathcal{F}}S = \delta_{\mathcal{F}}T$. Let S', T' be such that $S\rho \rightarrow^* S'\rho \not\rightarrow$ and $T\rho \rightarrow^* T'\rho \not\rightarrow$. According to the induction hypothesis of (S), the followings hold.

$$\begin{aligned} AS &\vdash \delta_{\mathcal{F}}\tau.S = \delta_{\mathcal{F}}\tau.S', \\ AS &\vdash \delta_{\mathcal{F}}\tau.T = \delta_{\mathcal{F}}\tau.T'. \end{aligned}$$

The inductive proof of (S) can be reiterated to show $AS \vdash \delta_{\mathcal{F}}\tau.S' = \delta_{\mathcal{F}}\tau.T'$. Hence $AS \vdash \delta_{\mathcal{F}}\tau.S = \delta_{\mathcal{F}}\tau.T$. □

It is a small step from Lemma 50 to the completeness result.

Theorem 51. *Suppose S, T are finite π-terms. Then $S = T$ if and only if $AS \vdash \tau.S = \tau.T$.*

Proof. Let $\mathcal{F}$ be $gn(S \mid T) \cup fv(S \mid T)$ and let $\{\varphi_i\}_{i\in I}$ be a complete disjoint partition $\mathcal{F}$. Then $S = T$ if and only if $\varphi_i S = \varphi_i T$ for every $i \in I$. For each $i \in I$, $\varphi_i S = \varphi_i T$ can be turned into an equality of the form $\varphi_i^{=}(\varphi_i^{\neq}S')\sigma_{\varphi_i^{=}} = \varphi_i^{=}(\varphi_i^{\neq}T')\sigma_{\varphi_i^{=}}$. Using Lemma 32 and Lemma 49, this equality can be simplified to $\delta S'' = \delta T''$, where $\delta S''$ and $\delta T''$ are complete normal forms. We are done by applying Lemma 50. □

5.3 Axiom for Box Equality

According to Theorem 42, a proof system for the box equality on the finite π-terms is the same as a proof system for the testing equivalence on the finite π-terms. De Nicola and Hennessy [19] have constructed an equational system for the testing equivalence on the finite CCS processes. Built upon that system, Boreale and De Nicola [9] have studied the equational system for the testing equivalence on the finite π-processes. So there is not much novelty about an equational proof system for the box equality on the finite π-terms. It would be however instructive in the present framework to give an outline of the proof technique that reduces the completeness for the box equality to the completeness for the absolute equality.

Since $\simeq \subseteq =_{\Box}$, one may devise an equational system for the latter by extending the system given in Fig. 4. The additional axioms are given in Fig. 5. These laws are the well known axioms for the testing equivalence of De Nicola and Hennessy [19] adapted to the π-calculus. It is well known that they subsume Milner's τ-laws (See Fig. 6). Let AS_b be the system $AS \setminus \{C1, C2\} \cup \{N1, N2, N3, N4\}$. It is routine to check that AS_b is sound for the box equality. To prove that the system is also complete, we apply the following strategy.

N1	$\lambda.S + \lambda.T$	$=$	$\lambda.(\tau.S + \tau.T)$
N2	$S + \tau.T$	$=$	$\tau.(S + T) + \tau.T$
N3	$n(x).S + \tau.(n(x).T + R)$	$=$	$\tau.(n(x).S + n(x).T + R)$
N4	$\overline{n}l.S + \tau.(\overline{n}m.T + R)$	$=$	$\tau.(\overline{n}l.S + \overline{n}m.T + R)$

Figure 5: Axioms for Box Equivalence.

T1	$\lambda.\tau.T$	$=$	$\lambda.T$
T2	$T + \tau.T$	$=$	$\tau.T$
T3	$\lambda.(S + \tau.T)$	$=$	$\lambda.(S + \tau.T) + \lambda.T$

Figure 6: Milner's Tau Laws.

$S =_\Box T$ if and only if there exist some S', T' such that $AS_b \vdash S = S'$, $AS_b \vdash T = T'$ and $S' = T'$.

The soundness of the strategy relies on the fact that AS_b is complete for $=$. To make the strategy work, we need to introduce a special set of π-terms, different from the complete normal forms, so that the box equality and the absolute equality coincide on these special π-terms.

Axiom N1 suggests that a summation may be rewritten to a form in which no two summands have identical non-tau prefix. For instance

$$AS_b \vdash a(x).S + a(y).T = a(z).(\tau.S\{z/x\} + \tau.T\{z/y\})$$

and

$$AS_b \vdash \overline{a}(b).S + \overline{a}(c).T = \overline{a}(d).(\tau.S\{d/b\} + \tau.T\{d/c\}).$$

Axiom N2 implies that either all the summands of a summation are prefixed by τ, or none of them is prefixed by τ. Moreover N1 actually says that one does not have to consider any τ-prefix immediately underneath another τ-prefix. Axioms N3 and N4 can be used to expand a summation $\sum_{i\in I} \tau.T_i$ to a saturated form. We say that $\sum_{i\in I} \tau.T_i$ is *saturated* if $\tau.(\lambda_1.T_1 + T_j)$ is a summand of $\sum_{i\in I} \tau.T_i$ whenever there is a summand $\lambda_1.T_1$ of T_i such that for every summand $\lambda_2.T_2$ of T_j it holds up to α-conversion that $\lambda_2 \neq \lambda_1$. These observations lead to the following definition.

Definition 52. *Let $\mathcal{F}$ be $gn(T) \cup fv(T)$. A finite π-term T is a* box normal form *if it is in one of the following two forms:*

1. *There are $A, B, C \subseteq_f \mathcal{N}$, where $C \subseteq B$, such that T is of the shape:*

$$\sum_{a\in A} a(x).\left([x\notin\mathcal{F}]T_a + \sum_{n\in\mathcal{F}} [x{=}n]T_a^n\right) + \sum_{b\in B}\sum_{n\in N_b} \overline{b}n.T_b^n + \sum_{c\in C} \overline{c}(d).T_c \tag{25}$$

   where $N_b \subseteq_f \mathcal{N} \cup \mathcal{N}_v$; and moreover the following properties hold:

   (a) *for all $a \in A$, T_a is a box normal form on $\mathcal{F} \cup \{x\}$;*

   (b) *for all $a \in A$ and all $n\in\mathcal{F}$, $x \notin fv(T_a^n)$ and T_a^n is a box normal form on $\mathcal{F}$;*

   (c) *for all $b \in B$ and all $n\in N_b$, T_b^n is a box normal form on $\mathcal{F}$;*

   (d) *for all $c \in C$, T_c is a box normal form on $\mathcal{F} \cup \{d\}$.*

2. *There is some finite set I such that T is of the following form*

$$\sum_{i\in I} \tau.T_i \tag{26}$$

   such that the following properties hold:

- *For each $i \in I$, T_i is a box normal form on $\mathcal{F}$ of the shape (25);*
- *$\sum_{i\in I} \tau.T_i$ is saturated.*

In the above definition we have ignored the conditionals. This is rectified in the next definition.

Definition 53. *T is a* complete box normal form *if $T \equiv \delta_{\mathcal{F}} T'$ for some box normal form T' such that $gn(T') \cup fv(T') \subseteq \mathcal{F} \subseteq_f \mathcal{N} \cup \mathcal{N}_v$.*

We can now state a lemma that correlates Lemma 49.

Lemma 54. *Suppose T is a finite π-term and φ is complete on a finite set $\mathcal{F} \supseteq gn(T) \cup fv(T)$. Then $AS_b \vdash \varphi T = \varphi^{=} \delta T'$ for some complete box normal form $\delta T'$ on $gn(\delta T') \cup fv(\delta T')$.*

Proof. The proof is a modification of the proof of Lemma 49 with additional rewriting using N-laws. □

We could have a lemma that parallels Lemma 50. But the following result is more revealing.

Lemma 55. *Suppose S, T are complete box normal forms on the finite set $\mathcal{F} = gn(S \mid T) \cup fv(S \mid T)$. Then $S =_\Box T$ if and only if $S = T$.*

Proof. Suppose $S =_\Box T$. By Lemma 54 we may assume that $S \equiv \delta S'$ and $T \equiv \delta T'$. Let ρ be an assignment that agrees with δ. Then $S'\rho =_\Box T'\rho$. In view of Lemma 33, we only need to show that $S'\rho = T'\rho$. So let $\mathcal{R}$ be the relation

$$\{(P, Q) \mid P =_\Box Q, \text{ and } P, Q \text{ are box normal forms}\}.$$

There are three crucial properties about this relation.

1. If P is of type (26) and Q is of type (25), then $P'\mathcal{R}Q$ whenever $P \xrightarrow{\tau} P'$. This is so simply because Q cannot do any τ-action.
2. If both P and Q are of type (26), then $P \xrightarrow{\tau} P'$ implies $Q \xrightarrow{\tau} Q'\mathcal{R}P'$.
3. If both P and Q are of type (25), then $P \xrightarrow{\lambda} P'$ implies $Q \xrightarrow{\lambda} Q'\mathcal{R}P'$.

For the detailed proofs of these claims, the reader is advised to consult [19, 9]. So $\mathcal{R}$ is a π-bisimulation. □

Theorem 51, Lemma 55 and the fact that complete box normal forms are complete normal forms immediately imply the completeness of AS_b.

Theorem 56. *Suppose S, T are finite π-processes. Then $S =_\Box T$ if and only if $AS_b \vdash \tau.S = \tau.T$.*

It is remarkable that axioms N1 through N4 actually reduce the box equality on the finite terms to the absolute equality on the finite terms. This is yet another support to the branching style bisimulation.

5.4 Remark

In theory of CCS, Milner's equational systems for the strong and the weak congruences on the finite CCS-processes [40, 56] and De Nicola and Hennessy's system for the testing congruence are well known. A complete system for the branching congruence was given van Glabbeek and Weijland [101] in the first paper on branching bisimulation, in which the following single tau law is proposed.

$$\lambda.(\tau.(S + T) + T) = \lambda.(S + T). \tag{27}$$

The axiom (27) is clearly equivalent to the combination of C1 and C2. Their proof of the completeness in [101] makes use of a graph rewriting system. Later [96] gave a more traditional proof of the completeness theorem.

The extension of these systems to the value-passing framework is not trivial, the reason being that every value-passing calculus is built on top of an oracle domain, say $\mathfrak{D}$. Early inference systems studied by Hennessy and

Ingólfsdóttir [33, 34, 35] are based on concrete semantics. An uncomfortable rule in all these systems, from the point of view of a proof system, is the so-called ω-data-rule (28).

$$\frac{\forall v \in \mathfrak{D}.S\{v/x\} = T\{v/x\}}{a(x).S = a(x).T} \tag{28}$$

A significant step was made by Hennessy and Lin [36] that introduces a whole new approach to the study of the value-passing calculi. The symbolic semantics dispenses with (28) by introducing a strong logic. One could argue that this use of a logic is cheating because it essentially makes use of a universal quantification operator that ranges over the oracle domain $\mathfrak{D}$. But the virtue of the symbolic approach is that one could define a value-passing calculus using a moderate logic that makes a lot of sense from a programming point of view, and then works out the observational theory with the help of the logic. Using this idea Hennessy and Lin [37] propose several symbolic proof systems for finite value-passing processes. A survey of the symbolic approach is given in [44].

For the name-passing calculi, the study on the proof systems was initiated in the pioneering paper of Milner, Parrow and Walker [60]. Their system is complete for the strong early equivalence. It contains the ω-name-rule (29), which is a variant of (28).

$$\frac{\forall n \in \mathcal{N}.S\{n/x\} = T\{n/x\}}{a(x).S = a(x).T} \tag{29}$$

In the presence of the finite branching property, (29) is more manageable than (28) since only a finite number of the premises of (29) have to be verified. A beautiful alternative to *rule* (29) is set of *axioms* for match/mismatch introduced by Parrow and Sangiorgi [72], among which the following one, S5, plays an indispensable role.

$$[x{=}y]T + [x{\neq}y]T = T \tag{30}$$

Axiom (30) offers the possibility to carry out case analysis within an equational system. It is obvious from (30) that Parrow and Sangiorgi's systems are only good for the π-calculi with the mismatch operator. The mismatch is also necessary to state the following law, S6, which first appeared in [72].

$$n(x).S + n(x).T = n(x).S + n(x).T + n(x).([x{=}y]S + [x{\neq}y]T) \tag{31}$$

Axiom (31) characterizes the *atomic* nature of interactions. The symbolic approach to proof systems however makes use of neither (29) nor (31). Lin's symbolic proof systems [45, 49] are capable of dealing with calculi with or without mismatch operator. Instead of (31), the systems in [45, 49] resort to a less attractive rule (32).

$$\frac{\sum_{i\in I} \tau.S_i = \sum_{j\in J} \tau.T_j}{\sum_{i\in I} a(x).S_i = \sum_{j\in J} a(x).T_j} \tag{32}$$

A proof system for the strong open bisimilarity is given in [83], in which all axioms are indexed by distinctions. The system without using distinctions is proposed in [29]. Since the following law, M5,

$$[x{\neq}y]\pi.T = [x{\neq}y]\pi.[x{\neq}y]T \tag{33}$$

is invalid in the open semantics, axiom (34) is proposed in [29] as a substitute for (33).

$$(a)C[[x{=}a]T] = (a)C[\mathbf{0}] \tag{34}$$

It is worth remarking that (34) is equivalent to $(a)C[[x{\neq}a]T] = (a)C[T]$ in the presence of (30). So it is enough even for the π-calculus with the mismatch operator.

The first complete proof system for the weak congruence on finite π-processes is Lin's symbolic system [46, 48]. The nonsymbolic systems were proposed by Parrow [70] using (32), and by Fu [29] using (31). In [29] the authors have also discussed complete equational systems for the weak open congruence. It is revealed that the open bisimilarities, as well as the quasi open bisimilarities [24], are quite complicated in the presence of the mismatch operator. It has been suggested that the complications are due to the introduction of the mismatch operator. The real culprit is however

the confusion of the names and the name variables. Study on the weak open systems pointed out that Milner's three τ-laws are insufficient [29]. An additional τ-law

$$\tau.T = \tau.(T + \varphi\tau.T) \tag{35}$$

is necessary. Notice that (35) is derivable from (33).

All of these tiny discrepancies appear a little confusing. What the present paper reveals is that all these incompatibilities disappear once the names are treated properly. Moreover, since the mismatch operator comes hand in hand with the match operator, there is no real interest in systems without the mismatch operator. Proof systems for π-variants can be obtained by extending AS with additional laws. For example the following two axiom schemes are introduced in [105].

$$\overline{n}(c).C[\overline{c}m.T] = \overline{n}(c).C[\mathbf{0}], \tag{36}$$

$$\overline{n}(c).C[c(x).T] = \overline{n}(c).C[\mathbf{0}]. \tag{37}$$

It is pointed out by Xue, Long and Fu [105] that $AS \cup \{(36)\}$ is complete for π^L and that $AS \cup \{(37)\}$ is complete for π^R.

A more challenging issue is to construct proof systems for the regular processes [55]. A regular process is not necessarily finite state [56]; it is generally finite control [48]. Milner addressed the issue in the framework of CCS. His strong complete proof system [55] and weak complete proof system [57] make crucial use of the following fixpoint induction rule

$$\frac{F\{E/X\} = E}{E = \mu X.F} \; X \text{ is guarded in } F. \tag{38}$$

Milner's approach has been applied to branching congruence by van Glabbeek [96]. It has been extended and applied to the value-passing calculi by Hennessy, Lin and Rathke [38, 76, 39] and to the π-calculus by Lin [46, 48]. All these more complicated complete proof systems are of a symbolic nature. The side condition of the fixpoint induction (38) renders the rule truly unwelcome. But as Sewell has proved in [88, 90] there is no finitely axiomatizable complete system for the finite controls. In order to derive the equality

$$\mu X.a.X = \mu X.\underbrace{a.\cdots.a}_{k>1}.X$$

from a finite system, one needs rule(s) in addition to axioms. It is possible to come up with a complete proof system in which all the rules are unconditional [89]. But it would probably not pay off when it comes down to implementation. Now if we have to stick to the fixpoint induction, how should we make use of it? Milner's straightforward answer [55], adopted in almost all follow-up work, is to introduce process equation systems. To apply this approach to the π-calculus, it is convenient to use abstractions over names [39, 48].

The finite states/controls raise the question of divergence. The axiomatic treatment of divergence has been borrowing ideas from domain theory [3]. Scott's denotational approach is too abstract to give a proper account of interactions, and in the case of divergence, non-interactions. Early treatment of divergence in process algebra is more influenced by the domain theoretical approach [19, 102]. It appears that the first successful operational approach to divergence is achieved in Lohrey, D'Argenio and Hermanns' work [50, 51] on the axioms for divergence. Crucial to their approach is the observation that all divergence of a finite control is due to self-looping. The Δ-operator, defined below, is isolated to play a key role in their inference systems.

$$\Delta(T) \stackrel{\text{def}}{=} \mu X.(\tau.X + T) \tag{39}$$

Lohrey, D'Argenio and Hermanns' systems consist of the laws to convert divergence from one form to another so that the fixpoint induction can be applied without any regard to divergence. One axiom proposed in [50, 51] is

$$\Delta(\Delta(T) + T') = \tau.(\Delta(T) + T'). \tag{40}$$

The law (40) is sound for the termination preserving weak congruence. But it fails to meet the codivergence property. Based on Lohrey, D'Argenio and Hermanns' work, Fu discusses the axioms for the codivergence in the framework of CCS [26]. The codivergent version of (40) for example is

$$\Delta(\Delta(T)) = \Delta(T). \tag{41}$$

It is worth remarking that (41) is valid for the strong equality. The proof systems using the Δ-operator begin to unveil the rich structure of divergence. It also provides a purely equational characterization of the internal actions. The codivergence satisfies another equational axiom:

$$\Delta(S + \tau.\Delta(S + S')) = \Delta(S + S'). \tag{42}$$

It is interesting to notice the similarity between (42) and (27).

The above discussion is meant to bring out the following point. The system AS plus the axioms of codivergence proposed in [26] give rise to a complete proof system for the absolute equality on the regular π-processes, which is more accessible than the symbolic proof system.

How about a complete equational system for the box equality on the regular π-processes. A crucial step would be to prove for the regular π-processes a property corresponding to the one stated in Lemma 55. It could turn out that this problem is far more difficult than one would have perceived. Rensink and Vogler [77] point out a surprising fact that Milner's fixpoint induction fails for the fair/should testing equivalence. Consider the process equation

$$X \quad = \quad a.X + a.\mu Z.(a.Z + b). \tag{43}$$

It is not difficult to see that

$$a.\mu Z.a.Z + a.\mu Z.(a.Z + b) =_{\Box} a.(a.\mu Z.a.Z + a.\mu Z.(a.Z + b)) + a.\mu Z.(a.Z + b).$$

The solution

$$RV_0 \quad \stackrel{\text{def}}{=} \quad a.\mu Z.a.Z + a.\mu Z.(a.Z + b) \tag{44}$$

is not box equal to the canonical solution

$$RV_1 \quad \stackrel{\text{def}}{=} \quad \mu X.(a.X + a.\mu Z.(a.Z + b)) \tag{45}$$

to the equation (43). A context that tells apart the processes (44) and (45) is $(ab)(\_ \,|\, \mu Y.(\overline{a}.Y + \overline{b}.d))$. This is because $(ab)(RV_1 \,|\, \mu Y.(\overline{a}.Y + \overline{b}.d))$ is strongly observable whereas $(ab)(RV_0 \,|\, \mu Y.(\overline{a}.Y + \overline{b}.d))$ is not.

Unlike the pure algebraic view of process [5], we tend to think of AS and AS_b as proof systems that help derive process equalities. This explains the particular statements of Theorem 51 and of Theorem 56. The emphasis on equational proof systems rather than on axiomatic systems has the advantage that the introduction of congruence relations can be avoided. What is stated in Theorem 51 is called promotion property in [29]. It is pointed out by Fu and Yang [29] that this property is absolutely necessary to prove that an algebraic system of the π-calculus is complete, the reason being that Hennessy Lemma [56] fails for the π-calculus [29].

6 Future Work

The studies in process calculi over the last thirty years largely fall into two main categories:

I. The first is about the diversity of models. A lot of process calculi have been designed and discussed [64].

II. The second is about equivalence. Many observational equivalences and algebraic equalities have been proposed, axiomatized and compared [97, 99].

In literature there is only a small number of papers that deal with expressiveness or completeness issues [14, 15, 67, 30, 28]. If we understand the situation correctly, we are at a stage where we try to seek the *eternal truth* in process theory [1]. It is our opinion that we will not be able to go very far if we do not seriously address the issue of *problem solving* with process calculi. It is only by applying a model to tackle real problems can our treatment of the observational theory of the model be verified. Talking about problem solving, there is no better process calculus than the π-calculus to start with such an investigation. What is achieved in this paper is a condensed account of the foundational theory of the π-models that provides support for problem solving.

Eternal truths have to be model independent. They would not be very interesting if the models we are considering are too liberal and too diversified. This is why our presentation of the π-calculus has followed the general principles and methodologies of Theory of Interaction developed in [27]. The prime motivation of Theory of Interaction can be summarized as follows:

> There are two eternal relationships in computer science, one is given by the equality relation (absolute equality) between the programs (or processes) of a model, and the other is by the expressiveness relation (subbisimilarity) between the models. By using these two fundamental relations, one can then introduce a number of basic postulates that formalize the foundational assumptions widely adopted in computer science. Now let $\mathfrak{M}$ be the class of all models. The first postulate asserts that a model belonging to $\mathfrak{M}$ must be computationally complete.
>
> **Axiom of Completeness**. $\forall\, \mathbb{M} \in \mathfrak{M}.\ \mathbb{C} \sqsubseteq \mathbb{M}$.
>
> The Computability Model $\mathbb{C}$, defined in [27], is the minimal interactive extension of the computable function model. The Axiom of Completeness can be seen as a formalization of Church-Turing Thesis. It places a considerable constraint on the world $\mathfrak{M}$ of models.

In [27] it is shown that π^M is complete in the sense that it satisfies the Axiom of Completeness. In [105] the completeness of π^L, π^R and π^S is established. So it makes sense to talk about problem solving in all these π-variants. We remark that completeness in our sense is much stronger than the so-called Turing completeness [28]. Some variants of CCS [56] are Turing complete [14, 14], they are however not complete in our stronger sense [27].

What is done in this paper can be seen as a book-keeping exercise. New results are obtained and old results are assessed in a uniform formalism. Based on what is set up in this paper, the π-model can be studied in three main directions.

III. A theory of π-solvability that goes beyond the traditional recursion theory should be useful to understand the power of the name-passing calculi. It can be shown that a pseudo natural number generator is solvable in the π-calculus. But a genuine natural number generator is π-unsolvable. It would be useful to develop a theory of π-solvability and investigate the diagonal methods for tackling π-unsolvability. Other possible issues to look at are nondeterministic functions definable in π, the recursion theory of the π-processes (say the formulation of the s-m-n theorem, enumeration theorem, recursion theorem etc.).

IV. A comparative study of the complexity classes defined by the π-programs against the standard complexity classes [68] would be instructive for a better understanding of the algorithmic aspect of the π-calculus. Such a study may begin with a formulation of the class P_π of the problems decidable in the π-calculus in polynomial time in a way the class NP is defined in terms of the Nondeterministic Turing Machine Model, and then investigate the relationship between P_π and NP.

V. At a more applied level, one could try to develop a hierarchy of programming languages implemented on the π-calculus. Previous studies in the program theory of the π-calculus have not discussed anything about universal process for π. So a great deal more has to be learned before we are confident of using π as a machine model to implement a full fledged typed programming language.

One may question the practical relevance of these new research directions. After all the π-calculus, unlike the Deterministic Turing Machine Model, does not have a physical implementation. The rapid development of computing technology has actually provided an answer. The Internet is an approximate implementation of the π-calculus! It is not completely, as one might argue, a physical implementation. But it is the kind of computing environment for which a theory of the π-calculus might provide just the right foundation.

Acknowledgments

This work has been supported by the National Natural Science Foundation of China (grant numbers 60873034, 61033002). The authors would like to thank the members of BASICS for their interest and feedbacks. Especially they would like to thank Xiaojuan Cai and Chaodong He for proof reading the paper. Chaodong He has pointed out to us that our previous proof of Lemma 50 and the previous statement of Theorem 42 were mistaken. He actually showed us how to correct the mistakes. The authors would also like to thank Huan Long and Jianxin Xue for their discussions on the previous versions of the paper. Their comments have led to several improvements of the paper.

The first author would like to thank Huan Long and Jianxin Xue for sharing the interest in the π-variants π^L, π^R, π^S. Some of the problems raised in a previous version of the paper are resolved in [105].

References

[1] S. Abramsky. What are the Fundamental Structures of Concurrency? We still do not know. Electronic Notes in Theoretical Computer Science, pages 37–41, 2006.

[2] R. Amadio, I. Castellani, and D. Sangiorgi. On bisimulations for the asynchronous π-calculus. In *Proc. CONCUR'96*, volume 1119 of *Lecture Notes in Computer Science*, pages 147–162. Springer, 1996.

[3] R. Amadio and P. Curien. *Domains and Lambda-Calculi*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1998.

[4] J. Baeten. Branching bisimilarity is an equivalence indeed. *Information Processing Letters*, 58:141–147, 1996.

[5] J. Baeten and W. Weijland. *Process Algebra*, volume 18 of *Cambridge Tracts in Theoretical Computer Science*. CUP, 1990.

[6] C. Baier and H. Hermanns. Weak bisimulation for fully probabilistic processes. In *Proc. CAV'97*, volume 1254 of *Lecture Notes in Computer Science*, pages 119–130, 1997.

[7] J. Bergstra, J. Klop, and E. Olderog. Failures without chaos: A new process semantics for fair abstraction. In *Formal Descprition of Programming Concepts – III*, IFIP, pages 77–103. Elsevier Science Publisher, 1987.

[8] M. Boreale. On the expressiveness of internal mobility in name-passing calculi. In *Proc. CONCUR'96*, volume 1119 of *Lecture Notes in Computer Science*, pages 161–178, 1996.

[9] M. Boreale and R. De Nicola. Testing equivalence for mobile processes. *Information and Computation*, 120:279–303, 1995.

[10] M. Boreale, R. De Nicola, and R. Pugliese. Basic observables for processes. *Information and Computation*, 149:77–98, 1999.

[11] M. Boreale, R. De Nicola, and R. Pugliese. Divergence in testing and readiness semantics. *Theoretical Computer Science*, 266:237–248, 2001.

[12] G. Boudol. Asynchrony and the π-calculus. Technical Report RR-1702, INRIA Sophia-Antipolis, 1992.

[13] E. Brinksma, A. Rensink, and W. Vogler. Fair testing. In *Proc. CONCUR'95*, volume 962 of *Lecture Notes in Computer Science*, pages 313–327, 1995.

[14] N. Busi, M. Gabbrielli, and G. Zavattaro. Replication vs recursive definitions in channel based calculi. In *Proc. ICALP'03*, volume 2719 of *Lecture Notes in Computer Science*, pages 133–144, 2003.

[15] N. Busi, M. Gabbrielli, and G. Zavattaro. Comparing recursion, replication and iteration in process calculi. In *Proc. ICALP'04*, volume 3142 of *Lecture Notes in Computer Science*, pages 307–319, 2004.

[16] D. Cacciagrano, F. Corradini, J. Aranda, and F. Valencia. Linearity, persistence and testing semantics in the asynchronous pi-calculus. *Electronic Notes in Theoretical Computer Science*, 194:59–84, 2008.

[17] D. Cacciagrano, F. Corradini, and C. Palamidessi. Separation of synchronous and asynchronous communication via testing. *Electronic Notes in Theoretical Computer Science*, 154:95–108, 2006.

[18] X. Cai and Y. Fu. The λ-calculus in the π-calculus. *Mathematical Structure in Computer Science*, 21:943–996, 2011.

[19] R. De Nicola and M. Hennessy. Testing equivalence for processes. *Theoretical Computer Science*, 34:83–133, 1984.

[20] R. De Nicola, U. Mantanari, and F. Vaandrager. Back and forth bisimulations. In *Proc. CONCUR'90*, volume 458 of *Lecture Notes in Computer Science*, pages 152–165, 1990.

[21] R. De Nicola and F. Vaandrager. Three logics for branching bisimulation. *Journal of ACM*, 42:458–487, 1995.

[22] C. Fournet and G. Gonthier. The reflexive chemical abstract machines and the join calculus. In *Proc. POPL'96*. ACM Press, 1996.

[23] Y. Fu. Variations on mobile processes. *Theoretical Computer Science*, 221:327–368, 1999.

[24] Y. Fu. On quasi open bisimulation. *Theoretical Computer Science*, 338:96–126, 2005.

[25] Y. Fu. Theory by process. In *CONCUR 2010*, Lecture Notes in Computer Science, Paris, France, 2010.

[26] Y. Fu. Nondeterministic structure of computation. *To appear in Mathematical Structures in Computer Science*, 2015.

[27] Y. Fu. Theory of interaction. *Theoretical Computer Science*, accepted, 2015.

[28] Y. Fu and H. Lu. On the expressiveness of interaction. *Theoretical Computer Science*, 411:1387–1451, 2010.

[29] Y. Fu and Z. Yang. Tau laws for pi calculus. *Theoretical Computer Science*, 308:55–130, 2003.

[30] D. Gorla. On the relative power of calculi for mobility. In *Proc. MFPS'09*, volume 249 of *Electronic Notes in Theoretical Computer Science*, pages 269–286, 2009.

[31] C. He. Model independent order relations for processes. In *APLAS 2010*, volume 6461 of *Lecture Notes in Computer Science*, pages 408–423, 2010.

[32] M. Hennessy. *An Algebraic Theory of Processes*. MIT Press, Cambridge, MA, 1988.

[33] M. Hennessy. A proof system for communicating processes with value-passing. *Journal of Formal Aspects of Computer Science*, 3:346–366, 1991.

[34] M. Hennessy and A. Ingólfsdóttir. Communicating processes with value-passing and assignment. *Journal of Formal Aspects of Computing*, 5:432–466, 1993.

[35] M. Hennessy and A. Ingólfsdóttir. A theory of communicating processes with value-passing. *Information and Computation*, 107:202–236, 1993.

[36] M. Hennessy and H. Lin. Symbolic bisimulations. *Theoretical Computer Science*, 138:353–369, 1995.

[37] M. Hennessy and H. Lin. Proof systems for message passing process algebras. *Formal Aspects of Computing*, 8:379–407, 1996.

[38] M. Hennessy and H. Lin. Unique fixpoint induction for message-passing process calculi. In *Proc. Computing: Australian Theory Symposium (CAT'97)*, volume 8, pages 122–131, 1997.

[39] M. Hennessy, H. Lin, and J. Rathke. Unique fixpoint induction for message-passing process calculi. *Science of Computer Programming*, 41:241–275, 1997.

[40] M. Hennessy and R. Milner. Algebraic laws for nondeterminism and concurrency. *Journal of ACM*, 32:137–161, 1985.

[41] K. Honda and M. Tokoro. An object calculus for asynchronous communications. In *Proc. ECOOP'91*, volume 512 of *Lecture Notes in Computer Science*, pages 133–147, Geneva, Switzerland, 1991.

[42] K. Honda and M. Tokoro. On asynchronous communication semantics. In *Proc. Workshop on Object-Based Concurrent Computing*, volume 615 of *Lecture Notes in Computer Science*, pages 21–51, 1991.

[43] K. Honda and M. Yoshida. On reduction-based process semantics. *Theoretical Computer Science*, 151:437–486, 1995.

[44] A. Ingólfsdóttir and H. Lin. A symbolic approach to value-passing processes. In J. Bergstra, A. Ponse, and S. Smolka, editors, *Handbook of Process Algebra*, pages 427–478. North-Holland, 2001.

[45] H. Lin. Complete inference systems for weak bisimulation equivalences in the π-calculus. In *Proceedings of Sixth International Joint Conference on the Theory and Practice of Software Development*, volume 915 of *Lecture Notes in Computer Science*, pages 187–201, 1995.

[46] H. Lin. Unique fixpoint induction for mobile processes. In *Proc. CONCUR '95*, volume 962 of *Lecture Notes in Computer Science*, pages 88–102, 1995.

[47] H. Lin. Symbolic transition graphs with assignment. In *Proc. CONCUR '96*, volume 1119 of *Lecture Notes in Computer Science*, pages 50–65, 1996.

[48] H. Lin. Complete proof systems for observation congruences in finite-control π-calculus. In *Proc. ICALP '98*, volume 1443 of *Lecture Notes in Computer Science*, pages 443–454, 1998.

[49] H. Lin. Complete inference systems for weak bisimulation equivalences in the pi-calculus. *Information and Computation*, 180:1–29, 2003.

[50] M. Lohrey, P. D'Argenio, and H. Hermanns. Axiomatising divergence. In *Proc. ICALP 2002*, volume 2380 of *Lecture Notes in Computer Science*, pages 585–596. Springer, 2002.

[51] M. Lohrey, P. D'Argenio, and H. Hermanns. Axiomatising divergence. *Information and Computatio*, 203:115–144, 2005.

[52] M. Merro. *Locality in the π-Calculus and Applications to Object-Oriented Languages*. PhD thesis, Ecole des Mines de Paris, 2000.

[53] M. Merro and D. Sangiorgi. On asynchrony in name-passing calculi. *Mathematical Structures in Computer Science*, 14:715–767, 2004.

[54] R. Milner. A calculus of communicating systems. *Lecture Notes in Computer Science*, 92, 1980.

[55] R. Milner. A complete inference system for a class of regular behaviours. *Journal of Computer and System Science*, 28:439–466, 1984.

[56] R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.

[57] R. Milner. A complete axiomatization system for observational congruence of finite state behaviours. *Information and Computation*, 81:227–247, 1989.

[58] R. Milner. Elements of interaction. *Communication of ACM*, 36:78–89, 1993.

[59] R. Milner. The polyadic π-calculus: a tutorial. In *Proceedings of the 1991 Marktoberdorf Summer School on Logic and Algebra of Specification*, NATO ASI, Series F. Springer-Verlag, 1993.

[60] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes. *Information and Computation*, 100:1–40 (Part I), 41–77 (Part II), 1992.

[61] R. Milner and D. Sangiorgi. Barbed bisimulation. In *Proc. ICALP'92*, volume 623 of *Lecture Notes in Computer Science*, pages 685–695, 1992.

[62] V. Natarajan and R. Cleaveland. Divergence and fair testing. In *Proc. ICALP'95*, volume 944 of *Lecture Notes in Computer Science*, pages 648–659, 1995.

[63] U. Nestmann. What is a good encoding of guarded choices? *Information and computation*, 156:287–319, 2000.

[64] U. Nestmann. Welcome to the jungle: A subjective guide to mobile process calculi. In *Proc. CONCUR'06*, volume 4137 of *Lecture Notes in Computer Science*, pages 52–63, 2006.

[65] U. Nestmann and B. Pierce. Decoding choice encodings. In U. Montanari and V. Sassone, editors, *Proc. CONCUR'96*, volume 1119 of *Lecture Notes in Computer Science*, pages 179–194, 1996.

[66] C. Palamiddessi, V. Saraswat, F. Valencia, and B. Victor. On the expressiveness of linearity vs. persistence in the asynchronous pi calculus. In *Proc. LICS'06*, pages 59–68. IEEE press, 2006.

[67] C. Palamidessi. Comparing the expressive power of the synchronous and the asynchronous π-calculus. *Mathematical Structures in Computer Science*, 13:685–719, 2003.

[68] C. Papadimitriou. *Computational Complexity*. Addison-Wesley, Reading, MA, 1994.

[69] D. Park. Concurrency and automata on infinite sequences. In *Theoretical Computer Science*, volume Lecture Notes in Computer Science 104, pages 167–183. Springer, 1981.

[70] J. Parrow. On the relationship between two proof systems for the pi-calculus, 1999.

[71] J. Parrow. An introduction to the π-calculus. In J. Bergstra, A. Ponse, and S. Smolka, editors, *Handbook of Process Algebra*, pages 478–543. North-Holland, 2001.

[72] J. Parrow and D. Sangiorgi. Algebraic theories for name-passing calculi. *Information and Computation*, 120:174–197, 1995.

[73] I. Phillips. Refusal testing. *Theoretical Computer Science*, 50:241–284, 1987.

[74] B. Pierce and D. Turner. Pict: A programming language based on the pi calculus. In G. Plotkin, C. Stirling, and M. Tofts, editors, *Proof, Language and Interaction: Essays in Honour of Robin Milner*. MIT Press, 2000.

[75] L. Priese. On the concept of simulation in asynchronous, concurrent systems. *Progress in Cybernatics and Systems Research*, 7:85–92, 1978.

[76] J. Rathke. Unique fixpoint induction for value-passing processes. In *Proc. LICS '97*, IEEE Press, 1997.

[77] A. Rensink and W. Vogler. Fair testing. *Information and Computation*, 205:125–198, 2007.

[78] H. Rogers. *Theory of Recursive Functions and Effective Computability*. MIT Press, 1987.

[79] D. Sangiorgi. *Expressing Mobility in Process Algebras: First Order and Higher Order Paradigm*. PhD thesis, Department of Computer Science, University of Edinburgh, 1992.

[80] D. Sangiorgi. From π-calculus to higher order π-calculus–and back. In *Proc. TAPSOFT'93*, volume 668 of *Lecture Notes in Computer Science*, pages 151–166, 1993.

[81] D. Sangiorgi. Bisimulation for higher order process calculi. *Information and Comptation*, 131:141–178, 1996.

[82] D. Sangiorgi. π-calculus, internal mobility and agent-passing calculi. *Theoretical Computer Science*, 167:235–274, 1996.

[83] D. Sangiorgi. A theory of bisimulation for π-calculus. *Acta Informatica*, 3:69–97, 1996.

[84] D. Sangiorgi. On the origin of bisimulation and coinduction. *Transactions on Programming Languages and Systems*, 31(4), 2009.

[85] D. Sangiorgi and R. Milner. Techniques of "weak bisimulation up to". In *Proc. CONCUR'92*, volume 630 of *Lecture Notes in Computer Science*, pages 32–46, 1992.

[86] D. Sangiorgi and D. Walker. On barbed equivalence in π-calculus. In *Proc. CONCUR'01*, volume 2154 of *Lecture Notes in Computer Science*, pages 292–304, 2001.

[87] D. Sangiorgi and D. Walker. *The π Calculus: A Theory of Mobile Processes*. Cambridge University Press, 2001.

[88] P. Sewell. Bisimulation is not finitely (first order) equationally axiomatisable. In *Proc. LICS'94*, IEEE, pages 62–70, 1994.

[89] P. Sewell. *The Algebra of Finite State Processes*. PhD thesis, The University of Edinburgh, 1995.

[90] P. Sewell. Nonaxiomatisability of equivalence over finite state processes. *Annals of Pure and Applied Logic*, 90:163–191, 1997.

[91] B. Thomsen. A calculus of higher order communicating systems. In *Proc. POPL'89*, pages 143–154, 1989.

[92] B. Thomsen. *Calculi for Higher Order Communicating Systems*. PhD thesis, Department of Computing, University of London, 1990.

[93] B. Thomsen. Plain chocs — a second generation calculus for higher order processes. *Acta Informatica*, 30:1–59, 1993.

[94] B. Thomsen. A theory of higher order communicating systems. *Information and Computation*, 116:38–57, 1995.

[95] R. van Glabbeek. Linear time – branching time spectrum. In *Proc. CONCUR'90*, volume 458 of *Lecture Notes in Computer Science*, pages 278–297, 1990.

[96] R. van Glabbeek. A complete axiomatization for branching bisimulation congruence of finite-state behaviours. In *Proc. MFCS'93*, volume 711 of *Lecture Notes in Computer Science*, pages 473–484, 1993.

[97] R. van Glabbeek. Linear time – branching time spectrum (ii). In *Proc. CONCUR'93*, volume 715 of *Lecture Notes in Computer Science*, pages 66–81, 1993.

[98] R. van Glabbeek. What is branching time semantics and why to use it? In G. Paun, G. Rozenberg, and A. Salomaa, editors, *Current Trends in Theoretical Computer Science; Entering the 21th Century*, World Scientific, pages 469–479, 1994.

[99] R. van Glabbeek. Linear time – branching time spectrum (i). In J. Bergstra, A. Ponse, and S. Smolka, editors, *Handbook of Process Algebra*, pages 3–99. North-Holland, 2001.

[100] R. van Glabbeek, B. Luttik, and N. Trčka. Branching bisimilarity with explicit divergence. *Fundamenta Informaticae*, 93:371–392, 2009.

[101] R. van Glabbeek and W. Weijland. Branching time and abstraction in bisimulation semantics. In *Information Processing'89*, pages 613–618. North-Holland, 1989.

[102] D. Walker. Bisimulation and divergence. *Information and Computation*, 85:202–241, 1990.

[103] D. Walker. π-calculus semantics for object-oriented programming languages. In *Proc. TACS '91*, volume 526 of *Lecture Notes in Computer Science*, pages 532–547, 1991.

[104] D. Walker. Objects in the π-calculus. *Information and Computation*, 116:253–271, 1995.

[105] J. Xue, H. Long, and Y. Fu. Remark on some pi variants. 2011.